THE SPACE ALLOCATION PROBLEM

A Dissertation
Presented to
The Academic Faculty

by

Nirvik Saha

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Architecture, College of Design

Georgia Institute of Technology
August 2020

THE SPACE ALLOCATION PROBLEM

Approved by:

Dr. Dennis Shelden, Advisor
School of Architecture
Georgia Institute of Technology

Dr. Athanassios Economou
School of Architecture
Georgia Institute of Technology

Dr. Perry Yang
School of City Planning
Georgia Institute of Technology

Dr. Russell Gentry
School of Architecture and Civil Engineering
Georgia Institute of Technology

Dr. John Haymaker
Research Director
Perkins and Will

Date Approved: June 18, 2020

Dedicated to my parents, Tapasya and Debasis

ACKNOWLEDGMENTS

Dr. Dennis Shelden, my primary advisor, offered deep insights about problem-formulation and philosophized about a “cloud of algorithms” with “chained” functionality which were critical to the development of this work. It was a unique opportunity to interact closely with Dr. Shelden and I sincerely thank him for his subtle guidance and patience.

Dr. John Haymaker’s sage advice has helped in shaping this research, understanding research methodology, and developing applications for design professionals and researchers. Dr. Haymaker has taken the time to discuss domain-specific problems and enforced a rigor, which ensured the completion of this research. I would like to thank Dr. Haymaker for his guidance and the unique opportunity to work with him.

It was a great opportunity to work with Dr. Perry Yang’s Eco-Urban lab, which nurtures a comprehensive design process, coupling technology with normative design goals. Dr. Yang’s belief in the uniqueness of the domain of design is a key factor that informed this research. Dr. Russell Gentry introduced advanced manufacturing, fabrication techniques, and a systems-based approach, which influenced this research. Dr. Athanassios Economou’s critical examination of computation in architecture helped strengthen the arguments in this thesis. I sincerely thank the committee members for their guidance.

My parents, Tapasya and Debasis Saha gave me the courage to pursue this research. They have inspired me to survive and persist. I would like to extend my gratitude to Debashree Chakrabarti (Saha), my wife, who is a continuing source of encouragement and support. I humbly dedicate this dissertation to my family and my advisors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SAP	Space Allocation Problems
SAT	Space Allocation Technique
CAD	Computer-Aided Design
BIM	Building Information Modeling
GIS	Geographic Information System
FAR	Floor Area Ratio
FSR	Floor Space Ratio
MDP	Markov Decision Process
McMC	Markov chain Monte Carlo
IDF	Integrated Design Framework
PLUGS	Planning Urban Growth and Simulation

SUMMARY

In the domain of architecture and planning, the space allocation problem (SAP) is a general class of computable problems which is employed by numerous design processes to assist in the generation of spaces of a layout and simultaneously satisfy design objectives. The SAP has eluded automation due to combinatorial complexity and geometric intractability. This thesis describes a computational framework for solving the SAP across multiple scales and domains of the application using reinforcement learning algorithms that generate spatial solutions with optimal space-activity relations.

In this research, a broad range of computable problems are addressed across three scales of design processes, namely, space planning, site planning, and interactive networks of city blocks. This is achieved by identifying the role of SAP in generating the spatial output of a design process and compartmentalizing the SAP into computable tasks. Each task is mapped to a spatial model that consists of a set of geometric operations driven by optimization algorithms or numerical relations. These techniques are referred to as the space allocation techniques or SAT and developed as autonomous modules. Each spatial model invokes a specific set of SAT modules, in sequence, and the models can be connected to solve the desired SAP.

The spatial models are integrated into a framework after considering the exclusivity of the task accomplished by the models, common methods, data structures, and the flow of information between models. It is proposed that the spatial output of large design processes is approximated by creating *workflows* of connected elemental models. A workflow can be reused to solve project-specific design problems by updating the inputs such as site

boundary or project requirements and bylaws. The workflows support design exploration and provide iterative user interaction such that for a given problem, it is possible to study entirely different solutions, explore the downstream propagation of a design decision, generate alternatives or determine an exact solution. The workflow permits re-usability to evolve the design solutions over numerous similar projects. These features of the proposal lead to an explicit design process that helps in preserving the information regarding design decisions. The proposal provides a semi-automation environment that allows users to develop spatial solutions, interactively, where the geometric or topological inputs can be altered at runtime and the system generates the solution.

To evaluate this proposal, several features of a standard solution are identified to address their usability in design processes, the generalization of SAP across geometric and topologically variant problems, and the diverse scales of design processes. These features aid in the development of an alternative design environment where the potential for semi-automation is explored. The case studies and test-cases presented in this thesis illustrate the interaction between the designer and the software where the user can alter basic inputs on a spreadsheet or change the governing shapes at runtime and the workflow dynamically updates the internal organization of spaces and their activities.

A prototype, namely, Integrated Design Framework or IDF, is implemented to demonstrate the applications of the proposed computational framework. It is anticipated that the development of SAP will support several activities related to architectural practice and research including design in practice, analytical research, and the development/deployment of building systems and subsequent processes such as the design of mechanical systems and structural design.

CHAPTER 1. SPACE ALLOCATION PROBLEM (SAP)

The space allocation problem or SAP, traditionally, refers to the constrained generation of spaces to develop floor plans of buildings. Computational methods for solving the SAP have been researched for a long time, at least since Armor and Buffa (1964). The prior research since the 1960s demonstrates the use of geometric operations and various optimization techniques. The SAP-related research is summarized as questions (Jo and Gero, 1998) that pertain to (a) reducing the "complex and nonlinear" problem into a computable problem, (b) processing the "combinatorial explosion" that emerges from a generative process, and (c) devising an evaluation process based on multi-criteria objectives. The SAP-related methods generate spatial output from generic inputs without human intervention and address the scope for computation and artificial intelligence in architecture and planning. This aspect of automation distinguishes the research in SAP from typical computer applications (CAD/BIM/GIS) where the user draws primitive shapes and manually organizes building geometry, which may lead to sub-optimal solutions due to the space and time complexity of the problem. (Eastman, 1973, 2008; Galle, 1981; Mitchell, 1981; Liggett, 2000; Kalay, 2004; Star and Estes, 2008; Calixto and Celani, 2015, Nauata et. al., 2020)

In this research, it is proposed that SAP represents a common class of problems that occur in numerous design processes. Variations of this problem are applied to design processes in practice and research to systematically generate solutions for layouts, configurations of buildings, parcellation of sites and mixed-use zoning problems, etc. In

the following sections of this chapter, computable models of SAP are introduced (section 1.1) where the models are used to approximate spatial output expected from certain design processes. The objectives (section 1.2) are categorized as (a) identification of the role of SAP in design processes or classes of design problems (b) identification and standardization of their inputs (c) an integration framework. The spatial models are informed by space allocation techniques (SAT), which are optimization and geometry algorithms based on the hypotheses stated in section 1.3. Observable features of SAP models are proposed (section 1.4) to evaluate their efficacy in solving the problems and implementation in practice or allied research. These features are used to evaluate the SAT proposed in prior research, determine the scope for improvement, and develop novel methods to support the objectives of this research. The spatial problems identified as SAP are described in section 1.5. Finally, applications of the spatial models of SAP and the thesis structure are described in sections 1.6 and 1.7 respectively.

1.1 Organization of Computational Solutions: SAP, models, SAT

In this thesis, the spatial output of design processes are interpreted as SAP-related tasks that are mapped to computable models. The SAP is deconstructed as a set of computable tasks in design processes which are identified by determining (a) if the task can be accomplished with available resources to gather and process information or data required and (b) whether or not the task has tangible objectives or an underlying logic to map inputs to the output. Each computable task is addressed by devising a model that

generates the spatial output by mapping the configuration of spaces to a series of geometric operations such that the design objectives or constraints are met. The independent variables that govern the operations are iteratively altered using an optimization algorithm (stochastic gradient descent) to generate constrained spatial output. Collectively, geometric operations, design objectives or constraints, and optimization algorithms are considered as space allocation techniques or SAT which generate the spatial output.

The space allocation techniques or SAT are developed as a common set of methods, such that, depending on the problem, certain methods can be selected, organized into a computational pipeline, and bundled into a model. Or, a model addresses a specific SAP-related task by invoking a set of methods from a pool of SAT. Models can be considered as containers of SAT, where each model is developed independently as generic entities available to the user at runtime. Since each model is a self-contained unit of inputs and operations, it reduces the computational resources of time and memory. This autonomy permits the exploration of algorithms by using alternatives from a pool of SAT and allows the development of features for generalization and connectivity between the models. The generalization of each model is addressed by:

i. Standardized numerical and geometric inputs based on prior research and discussions with designers.
ii. Eliminating errors and inconsistencies that arise from processing geometric operations and adjusting the optimization mechanisms to ensure predictable output across a variety of inputs.

iii. Using principles of object-oriented programming, namely, classes, polymorphism, encapsulation, templates, abstract, etc, the code is developed such that the SAT can be re-used to serve models with varying topologies and objectives.

The SAP-related tasks or spatial-models can be interpreted as a sequence of constrained geometric operations, given by the SAT. This interpretation permits increments in the scale of problems or types of solutions being explored by adding geometric methods to the common SAT pool, where new functionality is connected to the optimization algorithms through the common data structures. The mapping of tasks to models facilitates the computation of spatial processes of design problems by organizing the elemental models into a framework that permits the flow of information between the models. The interconnected models can be used in various permutations to form workflows that emulate the spatial output generated by large design processes. In this thesis, the models are considered as fundamental building blocks of the SAP, or, the SAP is addressed by concatenating a set of models.

1.2 Aims and Objectives

This research proposes a generalized approach to performance-optimized space allocation that is applied to a broad range of problems in architecture and planning. This objective is pursued by formulating computable models of design processes at specific scales. The collective behavior of these models is manipulated by additional objectives of standardized inputs and integration framework that allows a computational process to

generate top-down design solutions across the widely disparate scales addressed in this research. Within these broad objectives that generalize problems in architecture and planning, more detailed objectives are defined at specific scales.

1.2.1 Problems

The classification of SAP is based on the scale of the problem. Three fundamental design processes are targeted, namely floor plan layouts, site planning, and interactive networks of city-blocks are described in the following sections. At each scale, the SAT is designed to generalize the topologically variant problem.

1.2.1.1 Problem 1: Space Planning (Floor plans or Building Layouts):

Space planning or floor plan design are typical architectural problems that exhibit the features of a computable problem. In these problems, a given boundary is compartmentalized into spaces and a designated activity is allocated to each space. Geometric and topological constraints govern the organization of space-activity formations (Homayouni, 2007). It is known to be an intractable problem due to the combinatorial complexity (Galle, 1981; Jo and Gero, 1998). Despite prior research, layouts are still drawn manually using CAD/BIM/GIS related software. Since space planning is a fundamental step in the design and construction of buildings, there is an interest in the automation of this process (Liggett, 2000).

1.2.1.2 Problem 2: Parcellation, Massing and Site Feasibility Study (Site Plan Scale)

The SAP addresses site-level planning and development by determining the zoning and configurations of buildings to maximize FAR, while constraining the spatial entities based on objectives of the problem that include bylaws, program requirements of building types, area requirements of activity types, etc. The constraints applied to site-planning problems allow a large number of alternatives from which optimal configurations are filtered based on the aforementioned objectives. Subsequently, the spatial solutions are evaluated based on performance measures such as solar gain, sky-view factor, etc. Naturally, computational processes enhance the evaluation of design alternatives by providing exhaustive solutions with minimum human intervention. The site planning processes result in the generation of building mass with stacked floor plates where a vertical allocation of activity types are used to assign activities to the floor plates taken collectively.

Parcellation of the site and building mass hosted by each parcel are connected by numerical relations whereas the site feasibility studies are constrained problems to locate building footprints on the site based on proximity relations and generate the massing models based on area requirements. The open space requirements of the site, circulation patterns, and allocating a gross activity to the set of floor plates taken collectively are common to site planning problems.

i. *Parcellation*: The parcellation of a site is achieved by a process of subdividing the site, which is similar to space planning, where the outer boundary is recursively

split. Certain patterns persist in both processes to accommodate performance criteria such as daylighting and circulation (Steadman et al., 2000). But the objective of site parcellation pertains to the creation of developable regions, the determination of a circulation network within the site, or exclude regions such as lakes, parks, unbuildable terrain, etc (Miao, et. al., 2018). The optimization of geometric operations preserves the area and dimensions of parcels for buildings rather than the adjacencies or orientation.

ii. *Massing*: A building mass is the volumetric representation of the cumulative area of stacks of floor plates placed in each parcel. It is computationally formulated as a procedural geometry problem that relies on the properties of the parcel on which it is situated. The numerical relations between the underlying curve of the parcel and the extruded building footprint are given by program requirements and bylaws such as setbacks, step backs, the permissible area, height constraints, etc. Typically, a building mass is based on well-documented typologies that persist due to the ease of construction and verified performance (Steadman, 2014). These patterns form the basis for developing numerous models for parcellation of a site and building-massing, governed by common optimization modules.

iii. *Site Feasibility Study*: The site feasibility study is a nested problem based on constraints of bylaws and program requirements where different types of buildings (footprints) are located on the site based on a range of dimensions and distance-matrix specified by the design requirements. Subsequently, a vertical stack of floor plates is generated from the footprints. Activities are allocated to the set of all floor

plates, taken collectively, to fulfill the area requirements. This is a three-dimensional allocation of activities which is constrained horizontally by the building type, and vertically by the elevation of the floor plate. This allocation problem is a modification of the planar type of allocation used in floor plan design.

iv. *Grouping of site-scale problem:* The parcellation-massing problems are grouped because they are interdependent exploratory processes, and the site feasibility study is an independent process based on constraint-satisfaction. Parcellation-massing is governed by geometric operations that subdivide a site into specific geometric configurations and a building is projected onto the parcel, procedurally, to meet constraints. The site feasibility study is driven by stochastic optimization to address the specifications regarding the separation distance between buildings, open space requirements, and activity allocation.

1.2.1.3 Problem 3: Interactive City-Block Networks (Large Scale Planning)

Survey-based research in urban design and planning provide appropriate fields and structure for problem-formulation that informs the development of generative techniques to configure large-scale layouts for optimal space-activity relations in an urban setting. Computable models are proposed where the correlation between spaces is parameterized and the variables are exposed to the user. The designers can use locally relevant data or experiment with hypothetical values as inputs to manipulate the correlations between

spaces and activities thereby modifying the characteristics of the built environment within the scope of the design problem being addressed.

1.2.2 Constraints from Standardized Input Formats

Constrained spatial models are considered as elemental building blocks of SAP because each model represents a specific SAP-related task. The constraints allow designers to control and manipulate the model where the constraints of the SAP are used by the optimization algorithms to guide the geometric operations and generate a spatial output. The constraints are devised such that they integrate multiple performance criteria of the layout. For instance, adjacency constraints between spaces are related to usage, services, and allied design processes for mechanical systems, structural design, landscape, etc.

An objective of this research is the development of standard input formats for the above space allocation problems from which constraints are internally generated. This will permit the generalization of the models across projects because the structure of this information can be parsed by the model and processed into appropriate data structures of constraints, consumed by the optimization mechanisms.

1.2.3 Integrated Framework

In the objectives, above, SAP is formulated for various design processes. Process-specific SAP is broken down into discrete models of constrained problems with specific input and output. These elemental models are assimilated into a framework to share information through common data structures and methods. This facilitates an approach to connect hierarchical processes that shape the built environment. It follows that the connected models simulate a sequence of design processes. The linked design processes facilitate systematic design exploration, project-specific solutions, and contextualization of a proposed scheme.

An integrated framework of connected components enhances the scope of design problems being addressed. It permits the use of workflows of concatenated components to model complex project-specific design processes. For instance, a workflow could generate a comprehensive model by emulating a top-down design process that ranges from large-scale planning to the floor plans of a building (Figure 44). Or the user may develop a tentative contextual study by anticipating the elements of a larger scale with minimal effort.

1.2.4 Summary of Aims and Objectives

The role of SAP is identified in design processes. It is broken down into computable tasks which are mapped to models that integrate user-input interfaces, geometric operations, and optimization algorithms. The models are used in conjunction to generate

the spatial output expected from the SAP. A schematic classification of the problems, their corresponding models, and operations are provided in Table 1.

Table 1. Aims and objectives: computable models of design processes

Design Process (Problems)	Model & Operations
Floor Plans	Constrained partition of a curve to generate spaces and determine appropriate circulation *Operations*: partition, organize, generate
Site Feasibility (Site Plan)	Constrained search to locate footprints of buildings on site, generate mass, floor plates and allocate activities. *Operations*: Locate footprints, generate floor plates, and organize activities.
Parcellation and Massing (Site Plan)	Constrained subdivision of the site; generate building mass typology, floor plates, and provide appropriate circulation. *Operations*: Generate parcels and building mass
Network Allocation (large-scale planning)	Using a graph model of the built form, optimize the activities (nodes) and circulation (edges). *Operations*: Locate spaces-sites, generate circulation between them, and allocate activities based on FSR.
Locate: determine the appropriate position in 3d coordinates of activities or shapes. Partition: Subdivision of a closed curve Organize: using constraints, activities are allocated to geometric form. Generate: the creation of geometric forms based on numerical requirements Connect: determine the circulation between activities or spaces.	

1.3 Hypotheses

It is proposed that the SAP represents a typical set of design problems in architecture and planning, the solution to which requires a computational approach to address the combinatorial complexity and geometric intractability. Domain-specific hypotheses to support the objectives defined in section 1.2 are:

i. A common class of linked design problems: the spatial output of numerous design processes are formulated as SAP and addressed by workflows of concatenated models to provide comprehensive solutions to design problems.
ii. Patterns of geometry and objectives: the geometric formations and objectives of design processes or tasks addressed by the SAP exhibit commonalities that can be used to provide general solutions.
iii. Embedded design structures: the computable models provide an opportunity to structure the information produced during the process of developing the solution.

1.3.1 A common class of linked design problems

It is hypothesized that SAP is a common class of problems in numerous design processes that can be defined as a method to generate the geometry of spaces and map an activity to spaces such that they satisfy constraints provided by the user. By formulating a common problem across scales and topology, design problems can be examined elegantly with a reduced need for geometric modules and computational resources.

The integration of spatial models into a framework is based on an examination of the features of the built environment, which, at multiple scales suggests a top-down execution from planning to building mass configurations and floor plans. The sequence of execution (Eastman et. al., 2004) anticipates the organization of data structures of spatial models, connected across three scales of design processes, namely, interactive city-blocks, site plans, and floor plans. At each scale, the design problem is shaped by interactions between activities that can be classified as an organization based on separation, grouping, proximities, and orientation, irrespective of their physical manifestation or geometric forms taken by the activity. In order to facilitate the flow of design information across multiple scales, the commonalities between the topological representation of the design problems are exploited and an integrated model is proposed to reconcile the distinction between processes that require specialization.

The thesis identifies common representations of the SAP and attempts to provide elegant solutions to multi-scale generative design problems by an integrated framework of models of SAP in design processes. The connected models of SAP will allow the generation of a sequential solution to the disparate tasks involved in the design process. The approach taken in this research draws on common features of the SAP across the scales (Table 1) while supporting specialization of the techniques deployed at these different scales of design to arrive at comprehensive solutions.

1.3.2 Patterns of geometry and objectives

While the common representation of the problem leads to topological solutions, the conversion to diverse physical forms of the spatial output is addressed by appropriate geometric methods and data structures. The geometric formations also exhibit commonalities and allow repetitive use of the methods to solve variations in forms and objectives across the three scales of the design problems addressed by the thesis.

The variations in geometric forms of design problems are the most common form of customization required in design practices, for instance, the shape of floor plates is rarely identical. Diverse shapes of spaces in a layout such as general polygons, curves, composite shapes, and open segments are addressed by generalizing the geometric operations.

It is proposed that scale-related variations in problems can be addressed by common optimization tasks and geometric patterns. Although the problems are governed by seemingly divergent objectives, the tasks exhibit similar tendencies, for instance, the attraction or separation between activities. Similarly, geometric operations used to generate floor plans by subdivisions can be mapped to parcellation of site plans. The constraints of scale are resolved by operator overloading (OOP techniques) allowing optimization algorithms and geometric operations to be retained across various scales of design problems. This modular approach significantly reduces the number of modules and allows a seamless flow of information through the integrated framework of models.

1.3.3 Explicit Design Process

(structuring design knowledge and information)

In design practices, the spatial output of design processes is typically achieved by composing primitives in CAD/BIM-based software. The changes in initial assumptions or subsequent decisions made during the design of buildings and urban design projects are not easily stored or represented in a format for general consumption. Consequently, at the end of the design process, the information used to generate the solution is lost, diluted or it remains obscure and design practices cannot fully utilize the collective knowledge.

The computational models, proposed in the thesis, represents the various states of transformations that lead to an architectural scheme. The computational framework is intended to be used as workflows of connected models where the selection of a model represents the intermediate decision that generates the spatial output. The entire workflow-based process is preserved along with the constraints in the executable document. This document may be re-used for a different problem, the workflow may evolve over several problems or it may be examined to review the steps taken in solving a problem.

The integrated framework facilitates an explicit design process by ensuring that consistent results are generated when the workflows are re-used. The standardization of user-input format, a generalization of geometric operations, and consistent optimization algorithms facilitate the preservation of information in the workflow.

1.4 Features of SAP-Models & Semi-Automation

While the SAT can be verified by examining the constraints and output, several features of an integrated framework are identified (Table 2) to evaluate spatial-models and enhance their applicability in the development of architectural schemes. The features target various aspects of model-manipulation concerning the generalization of operations development of constraints and their interaction with optimization algorithms. Interfaces are proposed to transform user inputs into constraints, internally, and these constraints are used to evaluate the spatial output. Thus, the designer is exposed to an intuitive format to provide user-inputs through spreadsheets. The interfaces eliminate the need for expertise to develop appropriate constraints or knowledge of internal algorithms used by the model.

The development of SAP-models that utilize the proposed features enhances the potential for semi-automation in the generation of architectural or urban design schemes. Semi-automation of a design process is performed by the sequential actions taken by the user and the spatial-models, where the interaction triggers the optimization process embedded in the model. Thus the models dynamically generate spatial output from the updated inputs provided by the user at runtime. Semi-automation may be considered favorable due to the need for experimentation in the design process where constraints and geometric forms are constantly altered based on non-computable parameters related to design sensibility (Parsuram et. al. 2011). The features (Table 2) provide the necessary criteria not only for a general solution to the SAP but creates the opportunity to develop a semi-automated design process.

Table 2. Features of Space Allocation Models

	Tracked Attributes	Explanation
1	Constraints	Can a wide variety of objectives be integrated?
2	Generalization	Are the solutions generally applicable to all geometric forms?
3	Optimization	Can a scalable algorithm be developed to resolve many *types* of constraints?
4	Input Interface	Types of information that can be input.
5	User Interaction	Can the optimization algorithms support user interaction that changes the objective at runtime? (design exploration)
6	Applications / Scaling	Can the solution be generalized with minimum modifications to other design processes?
7	Semi-Automation Framework	If multiple processes are supported, can it be integrated into a greater model to support human interaction?

1.5 Necessity: Generation-Evaluation Paradox

At present, numerous performance metrics of a given building layout are simulated and measured in the virtual space, including daylighting, sound propagation, cost incurred, views, micro-climate, sky-view factor, movement of people, carbon footprint, etc (Haymaker et. al., 2018). The analysis of design options through metrics derived from these performance criteria are used to develop recommendations and guidelines. Numerous

studies indicate that this methodology helps determine locally optimized solutions (Chang et. al., 2019; Rezzae et. al., 2020). It would seem that the optimal design of the built environment can be generated by taking measurements for a variety of conceivable options.

But there is a paradox. Recommendations based on analysis have a greater impact when the measurements ***precede*** the actual design process. However, the analysis is based on measurements of the design options. This is the classic paradox of causality. While measurements of performance criteria have been addressed by sophisticated simulation and analysis techniques, the constrained generation of design solutions has not advanced to an extent where it can be used in practice or research to solve a general problem. Currently, the use of computation in generating design options are limited to partial studies with limited re-usability due to the diversity of geometric and topological problems in architecture and urban design. The generative methods commonly used in practice and research, involve the mapping of predetermined geometric shapes to stochastic variables that generate a singular class of variations of a pre-determined structure rather than address the constraints of the design problems. Effectively, a linear process is used to generate random variations or enumerations from which a subset of spatial solutions is selected by external processes. Optimization processes such as those envisioned by March (1976) or Eastman (1973) are cyclical where the layout is iteratively improved over several iterations and the solution is achieved at the end of the cycle.

March L. (1976) envisioned an "*automated"* design approach based on graph algorithms, mathematical programming, and methods analogous to circuit design and VLSI layouts (March L. and Steadman P., 1974). The proposed methods (a) generate design

options, and (b) evaluate options based on environmental performance criteria. To March L. (1976), while proposing the generate-evaluate loop, the SAP represents a missing automation technique or method in architecture and planning where the generation of optimal layouts and configurations of buildings is driven by the optimization of performance metrics. The SAP is closely related to a design process because they utilize the same constraints as designers and recursively generates solutions that are indicative of the schemes generated by architects or planners. Consequently, the spatial output is generalizable to various design problems and require minimum supervision. The proposed research in generative techniques is based on SAP that occurs in design processes with the explicit aim of enhancing the scope of data analytics by an exhaustive exploration of the design space, and, which is sufficiently flexible to include user-interaction or rely on internal mechanisms to generate the spatial output.

1.6 Applications

The spatial models, proposed in the thesis, can be applied in practice to generate spatial output based on fundamental inputs used by designers rather than encoding extensive relations between geometric forms and their conditional operators. The spatial models of SAP resolve design requirements and generate layouts for subsequent filtration based on simulation to extract schemes with enhanced performance. This is achieved by a methodology where SAP workflows are used with relaxed constraints or hypothetical values to generate a vast range of variations whose performance is measured and analyzed,

which leads to recommendations that are input back into the spatial models of the SAP as constraints to generate optimal architectural schemes.

The generate-evaluate loop provides avenues for project-specific holistic design because prescriptive rules and bylaws cannot be scaled to accommodate every aspect of the built environment, consequently, global recommendations do not address conditions for local optimality of individual developments or groups of disparate developments in a city (Lynch K, 1982, Bettencourt and West, 2010). This problem has expressed itself in the design of modern cities (Jacobs J 1964, Lampugnani 2010). Based on section 1.2, an alliance between generative techniques and evaluation is necessary to determine the correlations between performance measures and geometric forms of spaces (Roudsari, 2012). It is proposed that the space allocation techniques and analytical tools are appropriate methods that generate locally optimal solutions because they provide appropriate recommendations and guidelines for decision-making in design problems. The potential of analytical tools is enhanced when a large set of design solutions are available for sampling. The generative solutions allow the analyst to evaluate a vast number of potential design solutions, reach conclusions, and provide recommendations to improve the design. To explore a vast range of design options that meet project requirements before environmental evaluation, spatial models are proposed, which range from space planning to site planning, large-scale zoning, and development of space-activity networks and addresses typical design constraints, building by-laws, and guidelines.

Apart from applications in constrained design problems, proposed space allocation models generate alternatives for problems with soft constraints, where the solutions are

allowed within a tolerance range. They permit fast and extensive design exploration which is required during the initial and intermediate phases (Gane and Haymaker, 2012). Automated design generation and testing capabilities allow an examination of numerous possible layout options to assess the merits or flaws of the options and arrive at an optimal solution.

The architectural layout is an input for associated activities including structural design, energy analysis, performance measurement, and cost estimation (Chonga et. al., 2009). The configuration of spaces in a layout influences the circulation of people and services. In practice, the layout is refined over many iterations based on the discussions among stakeholders, and engineers (Eastman et. al., 20004). An ability to generate exhaustive design options, and filter them using performance indicators can help in assessing the multi-disciplinary impacts of design decisions. The estimated cost of development is committed at an early stage (Duffy et. al., 1993) and these estimates play a vital role in the selection of a conceptual model for further development. (Ulrich and Eppinger, 2000). Since SAP techniques are based on spatial form generation in design processes and generate solutions based on fundamental standardized constraints, the uncertainties in early stages are reduced. This reduction in uncertainty will help reduce the cost of the redesign (Salonen and Perttula, 2005; Okudan and Tauhid, 2008).

1.7 Thesis Structure

It is proposed that SAP in design processes are composed of tasks where each task can be mapped to computable models that utilize tangible organizational patterns and generate spatial output expected from the design process. Each model is composed of input-interfaces to accept user-inputs and generate objectives which are used by optimization algorithms to guide geometric operations that lead to a generative framework. The proposed models are interconnected and generalized to operate on topological or scaled variations of design problems.

In prior research, discussed in chapter-2, design processes related to spatial form generation are reduced to computable problems. The prior research is classified based on their generative techniques. Using the features of SAP, the scope for improvement in existing techniques is identified and novel methods to achieve these features are proposed.

An explanation of the proposed SAT, namely, information processing, optimization, and geometric operations, is provided in chapter 3. The interaction between optimization and geometric operations is explained. Generalization of geometric operations and the application of constrained modules is discussed. Specialized graph-based methods used in the development of data-driven urban design models are explained.

Based on the objectives of this research identified in section 1.2, and the SAT (chapter 3), computational models of SAP are developed to assist design processes. It involves the identification of relevant constraints for standardization and utilizing the

techniques developed in chapter 4 to develop elemental models. The accompanying implementation software (IDF and PLUGS) to test the proposed models is introduced.

Using the features of SAP, the results of the proposed models are illustrated in chapter 5. The application of IDF/PLUGS are demonstrated using appropriate test cases and case studies based on typical problems in the practice of architecture and urban design.

In chapter 6, the potential to address a wide range of objectives and support to data-driven decision-making as well as the application of the models in research is discussed. A comparison with existing paradigms of generative techniques is provided.

The role of the SAP in generating the spatial output of design processes is described in chapter 7. The significance and implications of the research are discussed. Limitations of the proposed methods and scope for advancement in generative techniques are indicated.

CHAPTER 2. REVIEW OF PRIOR RESEARCH IN S.A.P.

In this chapter, the prior research in generative methods and optimization techniques that address SAP in architecture and urban design are classified (Table 3) based on problem-formulation and the geometric techniques adopted by the researcher. This chapter introduces the initial developments in SAP with graphs, the evolution of constraints, and rule-based systems (section 2.1). In the following sections of this chapter (sections 2.2-2.8), each classification of the SAP is discussed along with the specific components of the technique that contributed to this research. After discussing the existing techniques, the scope for improvement in prior research is identified (section 2.9) based on features introduced in section 1.4 (p 21). The hypotheses that lead to novel techniques are discussed in section 2.10. These techniques enhance and extend the SAP to various design processes, allow them to operate on various scales, and ensure generalization over scale, geometry, or topological requirements.

Table 3 – Classification of prior research based on generative techniques or SAP formulation and a sample of prior research.

SL	Problem-formulation	Sample of prior research
1	Varying Dimensions (Mathematical Programming)	Gero (1977); Mitchell et al (1976, 1981); Tate and Smith (1995)
2	Dual of a graph	Grason(1971); Eastman(1973), Pfeffercorn (1975); Shekhawat K (2019)
3	Cell -Grid Formulation	Y. Bozer, et al (1994); Lopes et al. (2014); Ipek Gürsel Dino (2016)
4	Graph-Grammar, Procedural	Parish Y.I.H., Muller P. (2001); Chen et al. (2008); Yang et al. (2013)

(Table 3 continued)

5	Subdivision and Dissection	Tam K (1991, 1992); Marson and Musse (2010); Katja Knecht (2010);
6	Physics-based Modelling	Michalek and Palampabros 2002; Homayouni, 2007, Schneider and Koenig, 2012
7	Urban forms and Networks	Aliaga et al. 2008; Vanegas et al. (2011, 2012); Peng et al. (2014, 2016)

2.1 Early Investigation of Computable Solutions

Prior research in this domain has been extensively reviewed by March (1971), Mitchell (1981), and Liggett (2000). Initially, the SAP was defined as a quadratic assignment problem where a set of activities are assigned to a set of spaces such that the cost of the flow of materials and goods is minimized (Armor and Buffa, 1964). This problem was re-formulated in the domain of architecture to generate building layouts where appropriate constraints were introduced. While there are many niche techniques, in this research classification of general generative techniques is presented.

2.1.1 Space Allocation Problem

Among the initial techniques, a logic-driven topological solution has persisted to compute the floor plan where a layout of spaces was represented by a graph, where the nodes and edges of this graph represented activities and circulation respectively (Grason,

1971; Eastman, 1973; Pfeffercorn, 1975). Using graph theory, an alternative layout is found by reassigning activities to the nodes and simultaneously reconfiguring the edges to generate circulation for the updated system. Matching and enumeration algorithms were employed (March 1976, Lei and Leewand, 1988) to generate enumerations of the topological structure and develop quantitative solutions to the layout problem. Using the topological representation of the layout, Tabor and Willoughby (1972) proposed algorithms for optimal circulation. Performance measures for subsequent evaluation were suggested to filter the solutions leading to an integrated approach with environmental design as a natural goal of computer modeling (March L, 1976).

Various attempts to solve the problem were thwarted by the combinatorial complexity where the number of possible arrangements for spatial entities was found to increase exponentially with increments in the inputs (Galle, 1981), which poses a significant challenge to an exhaustive enumeration of the graph. This makes it an NP-complete (non-polynomial time) problem (Jagielski and Gero, 2006). This is exacerbated by the need for simultaneous optimization of constraints. Due to the complexity of the problem, computational methods using stochastic optimization are advocated to solve the SAP (Eastman, 1973). The techniques for generalization of geometric forms did not evolve in the early stages of SAP research.

This visionary phase of research in architecture recognized that certain problems were not addressed by the prevailing theory of architecture and planning. It led to research in identifying the role of computation in design processes. The research techniques were based on an integration between analytical studies and generative systems. This expanded

the epistemological foundations of the domain to include algorithms, systems engineering, analysis, numerical methods, etc., which required a mathematical treatment of architectural problems (Keller, 2006).

2.1.2 Evolution of constraints & Application of Heuristics

Alexander (1964) discusses the challenges in organizing the information used in design processes. This was attributed to the complexity of the *context* (requirements of the environment). Alexander proposed that the *form* (model) must be decomposed into the *interaction* (correlation) between variables of the design problem. From the *context* (prior studies of similar environments), a designer can determine if two variables are correlated.

The attempts to organize the information led to a standardization of inputs and constraints for space planning design processes. Over time, research in SAP led to the identification and classification of several constraints. The constraints are minimum requirements imposed on the computational process and they allow a user to manipulate the generation of a layout. The primary constraints were classified as geometric and topological (Homayouni, 2007). The geometric attributes of a spatial component include area, dimension, proportion, and orientation. Topological constraints include proximity to the periphery, adjacencies, views, and circulation. Developing a layout that satisfies these constraints is a non-trivial task. It is a process of generating arrangement of spaces and evaluating them. Bereft of automation, the number of samples examined may not guarantee an optimal solution (Gero and Kazakov, 1977).

The development of optimization techniques for SAP is based on search processes based on heuristics, where, a technique such as genetic algorithms can be applied to a wide variety of formulations such as cells in a grid (Rodriguez et. al. 2013) or varying dimensions of a known configuration (Michalek et. al., 2002). The genetic algorithms are primarily used to address the adjacency relations or the distance between spaces. HEGEL developed by Akin O et al (1987), for instance, is a problem of organizing hierarchies of spaces that use heuristics (Figure 1 a). Genetic programming (Holland, 1975) when used to solve the SAP involve the development of objective functions to evaluate a given layout (Jo and Gero 1998). The evaluation can be based on criteria such as adjacency, orientation, or circulation distance. Heuristics treat the SAP as a search problem, which, over several iterations, attempt to combine favorable substructures of randomly generated layouts and generate a layout with desired attributes.

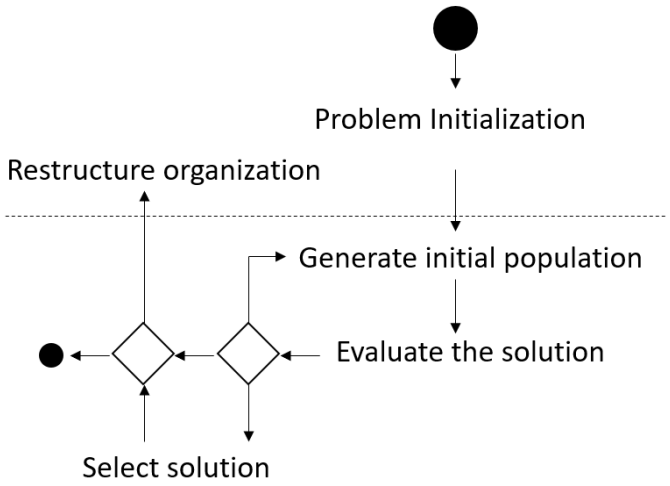


Figure 1 a. HEGEL uses heuristics to generate appropriately organized layouts.

(Figure 1 continued)

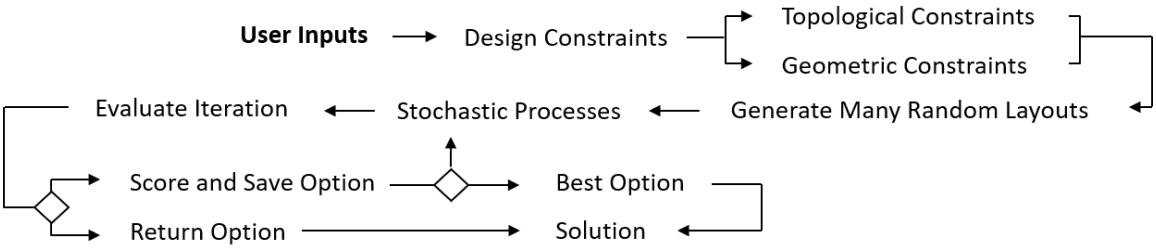


Figure 1 b. Typical optimization processes that rely on heuristics to generate an optimal solution.

Figure 1. A typical flow of information in heuristics

2.1.3 Rules and Shape Grammar

The early vision for generative systems in architecture and planning was a rule-based concatenation of primitive shapes. Alexander (1964) addressed a lack of structure in information being used by designers in design decision-making and proposed probability-based reasoning to organize elemental patterns required by the design problem. Perhaps, the most prominent catalogs of patterns of the built environment have been developed by Alexander et. al. (1977). Each pattern applies to a type of site, groups of people or community, their requirements, etc. Alexander proposed that these patterns can be concatenated to determine an appropriate living environment. This system was described as an algebra of patterns and rules of composing them are driven by the corpus or analysis of the contextual information. Alexander's theory is extended by Nagasaka (2020) who proposed the use of conditional probability theory to inform the concatenation process.

Similar generative systems are also proposed by March (1976, 1981, 1992) who argued that generative systems should be based on deductive, inductive, and abductive processes that, respectively, predict the performance of a component of the layout, assimilate it in a collection of components and synthesize a new layout from primitives. This was known as the PDI-model (production-deduction-induction) to synthesize and structure design knowledge.

Shape-grammar (Stiny G. 1975, 2006) is a method used by academicians in architecture where "rule-based programming" (Flemming U, 1989) is utilized to systematically reproduce design alternatives of a given scheme (Koenig and Schneider 2012). The methodology used to generate solutions is described by Flemming as consisting of two parts. First, by studying the architectural corpus, the relations between geometric forms of architectural entities are encoded as rules, which can be applied sequentially by detecting a rule and applying the transformation. Second, the rules are applied to generate a composition of shapes. The shape-grammar methodology of generating rules from the corpus and employing the rules with alterations to generate variations have been applied to create variations of the landscape of Mughal Gardens (Stiny and Mitchell, 1980), the urban planning of Marrakech (Duarte, 2005), the Palladian Villas (Grasl and Economou, 2010).

2.2 Varying Dimensions (Mathematical Programming)

Initial efforts in space planning involved the optimization of a "sketch" or an initial configuration of spaces provided by the designer. This sketch was a two-dimensional figure

which included the tentative location of each space. The constraints were provided as a range of dimensions and area requirements or objective functions, which are formulated as a set of inequalities. Solutions were facilitated by solving the inequalities using standard LP or MILP (March L, 1971; Gero, 1976; Mitchell, 1977; Peng et. al. 2016).

Encoding a designer's sketch into a computational form and optimizing the dimensions using mathematical programming provided salient features for the development of proposed solvers, used in this thesis:

i. Mathematical programming techniques are commonly used in scheduling, cost optimization, network analysis, etc. These techniques were introduced in architectural design in an attempt to solve the SAP.

ii. This formulation separated the geometric operations from the development of the topological structure of a layout because the initial configuration of spaces is input and the process updates the dimensions to achieve an optimal geometric solution. The separation of problems implies that once the topological structure is developed, the operations can be used to generate the appropriate spaces.

iii. The separation between topological and geometric optimization has been applied in many instances of the models proposed in this research. The topological optimization is addressed by using a representation of the tentative geometry. And the geometric attributes are generated after optimization of topological constraints. Essentially, the *sketch* is generated by topological optimization and geometric attributes of spaces are determined, subsequently by mathematical programming.

2.3 Graphs: circuit diagram, matching, and enumeration

Approaches based on graph theory have been proposed to develop a layout using the rectangular dual of a graph (Eastman, 1971; Grason, 1971). The layout is regarded as a directed graph with nodes as spaces and constraints as weights. The graph is altered to minimize the cost of traversal. Alternatively, matching algorithms have been proposed to construct graphs that represent the layout (Lei and Leewand, 1986). The most suitable graph is transformed back into a layout by taking the rectangular dual. It was inspired by circuit diagrams and VLSI layout planning. Graph-based approaches allow enumeration of possibilities by using the Burnside lemma or Cauchy-Frobenius theorem. In principle, the permutations of a plan can be subsequently filtered by adding constraints. The most common geometric shapes are rectilinear, and the vast number of possibilities makes it mathematically interesting. The following features of the prior research using graph-theory led to the development of the proposed solvers, used in this thesis:

i. Numerous layout problems are represented by a topological representation of the layout and this equivalent graph-based representation of the problem is exploited in developing the common optimization modules. Apart from space planning, the problem of interacting networks (section 1.2.3) is a variation of the graph-theoretic approach where the generative model is two-fold. First, the topological structure of the graph is configured. Then the geometry of spaces is developed. Constraints for the model are based on graph-theoretic measures commonly found in the study of urban characteristics.

ii. The graph-based representation of a layout is convenient for optimization algorithms. After the optimization process, the geometric structure may be reconstructed from the graph. This formulation allows scaling and many types of optimization algorithms can be applied to the layout in this form such as DFS, BFS, A-star, LP, etc.

iii. Using the graph-based representation, typical graph algorithms can be used. For instance, centrality and dispersal are used to distribute the nodes or activities, while, matching algorithms, greedy vertex cover, etc., are used to solve various allocation problems in SAP. The connectivity of edges is informed by the minimum spanning trees and shortest path algorithms.

2.4 Cell-Grid Formulation

In prior efforts, the occupation of cells in a grid is used to generate the geometry of floor plans. It is achieved by discretizing an enclosed region into shapes or cells. The cells are grouped to form spaces that satisfy desired constraints. This is a flexible approach that allows the application of mathematical programming, heuristics, and L-systems (Mitchell et al 1975; Liggett 1981; Jo and Gero, 1998).

Cell-based formulations can be likened to cellular-automata where rules are applied to cells taken in sequence. For instance, March (1976) proposed a boolean formula to govern the extrusion of cells and demonstrated Mies' design of the Seagram Tower. Similarly, Steadman (2014) demonstrates the application of cell-grid formulation of

layouts by developing a classification of building typologies and built form. Based on a survey of British hospital layouts, Steadman represented spaces and corridors as cells and proposed a boolean formula to fulfill daylighting requirements of spaces and describe their organization. The binary coding depicted the relation between rooms, corridors, and courts or light-wells inside the building.

Cellular-automata, an elaborate formulation of cell-occupation using rules, provide means to develop dynamic systems that demonstrate urban development over time (Batty 1997, 2003). The application leads to spatial and temporal behavior depicted by the occupation of cells from initial conditions and the repetitive application of rules. The artificial agents occupy cells based on hard-coded rules such as the density of nearby cells which represents aspects such as demand or rent and urban growth (Torrens and O'Sullivan, 2001; White and Engelen, 1993). Apart from spatial complexity, cellular automata are used in socio-economic models in planning (Torrens, 2010).

Formulation of SAP that uses a logic-based organization of discrete cells has persisted (Tutenel et al, 2010; Dino, 2016) as a generative technique for modest scale layouts because it can generate a variety of geometric forms and utilizes contemporary optimization solutions. A matrix of cells in a boundary is used as an input. These are split into zones. Circulation is placed between zones and defined as a requirement along with spaces. Initially, spaces are assigned randomly to cells. Each space is expanded in the direction of higher adjacency value based on the area requirements. Growth patterns may be pre-defined to include orthogonal shapes such as rectangles, L-shapes, T-shapes, or pre-

defined combinations of these shapes. This is a procedural technique using rules for building generation.

Discretizing the boundary into cells and occupying the cells is a common technique used to organize spaces in complex shapes while the attributes are constrained by numerical ranges. Since the discretization process generates an array of cells, the indices of each cell and associated geometric forms permit them to be used in association with a diverse set of optimization algorithms to determine properties such as the area of groups of cells or the length of traversal. The cell grid formulation can be developed using *any* process of discretization of the enclosed curve that allows consistent indexing of the generated discrete cells. Commonly, heuristics such as genetic algorithms (Calixto and Celani, 2015) are used to govern the cell growth and determine the appropriate locations of each space.

2.5 Rule-based, Graph Grammar and Procedural Models

Rule-based systems and graph-grammar are routinely used under the paradigm of parametric models to organize a non-trivial set of relations. The models encompass a wide range of applications from studies of rule-based concatenation 2d-shapes to a scaled generation of urban forms and facade systems (Parish Y.I.H., Muller P., 2001).

Apart from rules for concatenation of elemental forms or “patterns” suggested by Alexander et. al (1977), repetitive organization of geometric forms have been studied by Steadman et. al. (2000), where a classification of the built environment was proposed by

surveying 3,350 addresses across 4 English towns for the Department of Environment. This classification proposed by Steadman's study was based on internal subdivisions governed by daylighting. The research led to the extraction of rules for the organization of buildings on-site, void spaces (courtyards), and their extrusion. The study showed the existence of *"principal forms"* and *"parasite forms"*. The interior organization of buildings was primarily based on the propagation of natural daylight. They concluded that a peripheral band of activities and bays of double-loaded corridors are the most prevalent forms of organizing the layout of large buildings.

From the domain of computer graphics and games, examples of rule-based generative systems have been provided by Yang Y. L. et al. (2013), where pre-defined shapes of building footprints were used to occupy a closed curve (site) and generate urban forms. The template-based footprints are adjusted to fit the site using a set of predetermined rules. This method includes rules and procedural geometry to manipulate curves and polygon segments. Similarly, Peng et al. (2014) used tessellations to generate layouts of various shapes. This formulation was similar to a grouping of cells in a given bounded region but templates were enforced in this case to increase the efficiency of the system. The shape of the spaces was generated as a result of union operations and constrained solutions were obtained using mathematical programming.

Vanegas et al., (2010, 2011) propose user-driven inverse modeling systems. The pre-defined geometry of urban form is provided as a parametric model. The users can manipulate various constraints of the model to generate the desired form. The proposal is based on the hypothesis that urban planners (and content designers of computer games)

are aware of the local adjustments required to fit a given model to the site provided. The parameters included are the distance between intersections, road length & width, random rotations, etc. The model generated building heights based on sunlight exposure and floor area ratio. The following features influenced the proposed solvers, used in this thesis:

i. Graph-grammar replicate patterns of the topological structure and allow them to be synthesized to form new geometric formations using conditional operators. Rule-based systems implement an operation when certain conditions are satisfied. Both these techniques are related to the identification of patterns in the built environment and their concatenation. A trivial example is the triggering of a transformation when a shape is detected. For instance, Lindenmayer systems (1968) simulate the growth of a biological tree. Or, cellular automata are also considered instances of rule-based systems where probability-based reasoning and regression models of variables are utilized to simulate a physical process such as land-use, segregation of neighborhoods, activities in a city, rent-income-neighborhood studies in a district, etc.
ii. In this research, the proposed patterns are replicated by procedural geometry. It is a set of predefined geometric operations that are processed sequentially. They are used in conjunction with optimization algorithms. The direct application of these techniques was used to develop the proposed massing typologies. Apart from that, the proposed geometric operations are instances of procedural geometry.

2.6 Subdivision and Dissections

Perhaps a more recent technique for generating the geometry of floor plans is a "squarified-treemap*"* for room subdivision (Marson and Musse, 2010). This method is inspired by an algorithm used to visualize "hierarchical information structures" or the organization of data in a disk (Johnson and Schneiderman, 1991). In the domain of architecture and planning, this formulation recursively subdivides a rectangular boundary representing the floor plate curve to achieve a set of spaces that, collectively, form the desired layout. The subdivision process may be implemented hierarchically, where the subdivisions split the boundary into zones and these zones are subsequently subdivided into rooms.

Similar to the "squarified-treemap", the K dimensional-tree algorithm is used to control the subdivision process and generate a layout (Knecht and Koenig, 2010). is a process where a random set of points are scattered in a rectangular boundary and from each point, vertical or horizontal rays are drawn in opposite directions until they intersect the original boundary or a previously drawn line segment. This process is implemented recursively until the outer boundary is subdivided into the required number of spaces. Even though this method is restricted to rectangles, it generates a representation of the floor plan that is sufficiently realistic in the virtual environment for computer games.

Using typical architectural constraints, Tam (1999) proposed slicing tree algorithms to subdivide a rectangular boundary. A tree (graph with nodes and vertices) was generated from requirements. Rooms with high interaction were placed close to each other.

They formed the leaves of the tree. The subdivision was performed using the internal nodes of the tree. Stochastic variables were used to alter the internal nodes leading to variations in the subdivision.

In this thesis, subdivisions were used in the proposed solution to generate spaces inside a closed curve of the floor plan. It was generalized over curves and polygons. The binary structure allowed the development of the exact area and adjacency requirements. Q-learning with dynamic programming was used to find an exact solution in this thesis.

2.7 Physics-based Model

Physics-based models are proposed by prior research to provide interactive solutions where spaces are considered as point-mass and forces are assigned to attract or separate the spaces (Arvin and House, 2000). This formulation relies on initial sketches to generate a set of spaces and organize them into a planar configuration. Michalek et al, (2002, 2005) illustrate a layout with 23 spatial entities and associated relationships between them. It includes architectural constraints for dimensions, location of rooms, connectivity, and minimum wastage of space. Besides, the cost of heating, cooling, and lighting was minimized based on relative location. A genetic algorithm was used to resolve the topological relationships, where the arrangement was refined over iterations using randomness to find alternatives for low-performing design options.

In Physics-based models, the constraints of adjacency and separation are encoded into attraction and repulsive forces between spaces. A simulation model is implemented to run on a continuous loop where the states of the spaces are refreshed repeatedly and their location is determined by the application of the forces. The spaces are transformed by the forces until they reach equilibrium, or the velocities are minimized on detecting a collision between spaces. The following features of the prior research using Physics-based formulation led to the development of the proposed solvers, used in this thesis:

i. It is an alternative formulation where forces of attraction and repulsion between elements can be used to represent constraints of adjacencies, orientation, views, etc.
ii. The numerical cost calculations in layouts are commonly used in optimization calculations and balancing of forces is analogous to the optimality conditions achieved by the layout when conventional optimization techniques are used.
iii. User-interaction with models to readjust constraints at runtime enables a dynamic approach to determine layout solutions. In most prior solutions, the optimization process is static where the entire stack of computations must be completed before a reliable layout is generated.

2.8 Urban Forms and Networks

In the domain of computer games and graphics, procedural models are used to generate realistic urban forms that are comparable to satellite imagery. These techniques are specific to patterns and use procedural models to replicate design patterns. Procedural

methods for the generation of grids by designing tensor fields have led to successful results that resemble existing patterns (Chen et al, 2008). Aliaga et al. (2008) discuss the widespread use of the medial axis transform in typical urban forms such as parcels and streets. Vanegas et al. (2011) implemented modifications of the curve skeleton that allowed them to recreate forms of streets and parcels in a city. The prior research incorporated interactive methods where the user can locally alter fields and set various parameters for grid generation (Vanegas, 2012).

The network optimization problem is demonstrated by Peng et. al. (2016) where a network of roads and buildings is interpreted as a mesh to optimize circulation routes along the edges and organize interior spaces into mixed-use buildings. The quad-mesh was accepted as input along with edge-weights as traffic conditions to generate street networks and floor plans. The permissible shapes of building footprints and parcels were predefined and achieved as a result of union operations. This space generation is similar to grammar or rule-based systems that combine cells and treated the system of parcels and roads like a network optimization problem that is solved using integer programming. The rules were based on achieving rational shapes while optimizing the travel time and density of traffic.

Planning and urban design recommendations, developed by surveys or mapping, provide a strong basis for determining the correlation between variables (Straszhem, 1987; Friedman and Kern, 1997). They rely on network analysis to explain urban phenomena such as pedestrian movement (Hillier and Hanson, 1987), favorable location of retail (Sevtsuk, 2010), employment distribution (Waddell, 2003), etc. This information leads to the determination of attraction and separation in interactive networks where each network

consists of a certain type of activity (node) for instance residential or commercial. The efficiency of the organization of space-activity networks is evaluated using both the topological and physical structures to provide evidence for urban phenomena that can be corroborated by collecting data and developing regression models (Lynch 1981, Steinitz 2008, Dangermond 2012)

Spatial networks of the urban environment are studied in the form of a topological graph with nodes of activity connected by streets as edges. From this representation, various measurements are taken to determine the centrality of space (the focal point of pedestrian activity), attraction or sequential usage of spaces, preferred proximity between spaces, etc. In developing the "*Urban Network Analysis*", Sevtsuk and Mekonnen (2012), used the following metrics:

i. Numerical values for metric/topological *affinity* between networks/nodes similar to the adjacency matrix of space allocation.
ii. *Cost of travel* and reward for desirable activities reachable from a node (proximity matrix). (Bhat and Handy, 2002).
iii. *Gravity Index* (Hansen, 1959) determines the appropriate density or dispersal of certain activity types around others. It is controlled by rewards for reaching, and the cost of travel to a node based on distance from a node (center of gravity).
iv. *Centrality* is given by the node which is closest to all other nodes. It can be used as a metric to determine the topological distance from the center of the network.
v. A general measure for *betweenness* is the positive or negative values for a certain set of nodes between any two nodes or in a network.

Studies concerning network analysis have informed the proposed development of solvers used in this thesis concerning space-activity networks in city-blocks in the following ways:

i. The proposed techniques for generating urban forms and networks are based on a topological representation of space-activity networks, and the organization of input fields is based on the models studied in prior research.

ii. The geometry of the output as a result of network optimization is based on accepted patterns of urban development, specifically, building typologies and configuration of parcels are based on well-known patterns.

iii. The geometric techniques for large scale generation and optimization of street-grids are developed using the mesh-based tessellation techniques.

iv. The relations extracted from analytical studies are used to drive the configurations of networks of activities. The data provides numerical constraints for proximity or attractiveness between nodes of a topological representation of the built environment (zones). The relations extracted from analytical studies are used to drive the configurations of networks of activities.

v. Based on (iv), a generative model is developed as a generic graph that is shaped by inputs. These inputs correspond to the analytical measurements.

vi. Even in the absence of data from surveys, the models present a systematic generative solution. It is proposed that locally appropriate inputs may be used to test a hypothesis or generate alternatives.

vii. The topological structures along with typical graph algorithms and data structures commonly used in the analysis of urban characteristics were used in this research.

2.9 Scope for Improvement

The prior research demonstrates that a wide variety of design problems can be generated by computational methods. The features of SAP models identified in section 1.5 (Table 2, p. 22) are used to identify the scope for improvement or opportunities in each method to improve the solution. They are discussed in the following sections.

2.9.1 Constraints

Constraints are based on the needs of the design processes and determine the attributes to be optimized. While the constraints for generative models in space planning have been standardized over time, site planning and multi-block-planning process are not adequately addressed. Although the prior research (sections 2.4, 2.5 and 2.8) output solutions to site planning, or generation of street networks, the inclusion of architectural (domain-specific) constraints, used by designers, may introduce complexity to the problems, which will require additional processing.

2.9.2 Generalization of Geometry

The space allocation problem has been formulated with cells in a grid, rectangular dual of a graph, or the location of rectangles in a boundary. These methods employ simple shapes and allow the operation of various multi-objective optimization algorithms. However, layouts are composed of general polygons and curves (Fletcher and Cruishank, 1996). This requires sophisticated techniques for applications in practice and research.

2.9.3 Optimization

Eastman (1973) proposed the use of heuristics. In prior research, the combinatorial explosion (Galle, 1981) has been addressed with algorithms such as simulated annealing, and genetic algorithms among others (Calixto and Gelani, 2015). They rely on randomness to propagate parts of a solution where it is expected that over several iterations, erroneous sub-solutions will be eliminated. But optimal substructures of the layout are not easily joined over iterations (Sutton and Barto, 1998). Heuristics are also limited by scaling (Lighthill, 1973) and may not be suitable for large constraints of spaces and their relations. Stochastic optimization does not perform well in a continuously changing environment such as a design process where constraints are repeatedly altered.

2.9.4 Input interfaces

Since design requirements vary for each problem, an interface is required to enable the realtime processing of user inputs to dynamically generate objectives that manipulate the spatial output of the model. The development of an interface eliminates the need for additional expertise such that a designer can use the proposed models without a knowledge of the internal workings of the algorithms.

The interface allows a user to provide inputs to an intuitive application such as spreadsheets and processes the inputs to generate the objective functions. The pipeline is feasible when the format is standardized for inputs of geometric attributes, matrix-based relationships, orientation preference, etc. Apart from spreadsheets, intuitive GUI is used in this thesis when variables are exposed to the user. The GUI accepts inputs that take the form of numbers, geometric forms, boolean values, and strings.

2.9.5 User interaction

The input interfaces (section 2.9.4) facilitate the development of a robust system where the user may continuously update constraints. This has a cascading effect on the SAP model where the constraints are used by the optimization algorithms to guide the geometric operations. It requires the algorithm to halt and update the objective function, process substructures, and target sub-optimal spaces without disturbing the optimal partial configurations. The constant alteration of inputs and constraints during the development of

spatial output is natural in manual design practices but difficult to achieve with a computational solution because of the computational complexity of the optimization and geometric formulations of the SAP-model.

2.9.6 Applications and scaling

The prior research demonstrates that a wide range of design problems can be addressed by computational methods. The problems were developed independently across various domains spanning architecture, urban design and planning, computer graphics, and games. The application of the techniques provided by prior research can be improved to address the proposed objectives (section 1.2) in this thesis by:

i. Space planning (floor plans): While the prior research in space planning provides an exact set of constraints and demonstrates the application of optimization techniques, the research fails to generalize the problem across shapes or increments in scale with altered constraints. Whereas, generalized geometric techniques such as tessellation or interpolation are used without rigorous constraints and optimization techniques developed in the former case because architecturally appropriate forms are not targetted. The scope for improvement lies in problem-formulation that utilizes well-known computational geometry algorithms to generalize the topological variations.

ii. Site planning: section 2.5 provides techniques for generating a variety of geometric forms that resemble images and formations in the built environment but do not consider constraints that are required in practice or research. For application in practice of

architecture and urban design, bylaws and program requirements are necessary. This requires generative techniques with an appropriate evaluation mechanism in the optimization process.

iii. Interactive city-block networks: These are the most complex form of organization where entire networks are addressed. Typically, stochastically generated street networks and urban forms are used in computer games, which provide a realistic image but the constraints of urban networks are not considered whereas large-scale planning problems are informed by statistical tools (data analytics) which can be used in the decision-making process to determine the gross distribution of activities based on current performance (section 2.8) but these schemes do not address the actual forms of the networks or existing structures due to scaling problems.

2.9.7 SemiAutomation framework

By definition, the SAP can be applied to a wide variety of design problems by generalizing the operations for generating the geometric form and their respective optimization algorithms. The various techniques discussed in prior research demonstrate the possibility of a framework of solutions to address several design processes in architecture and planning. The aspects which may lead to a cohesive framework are consistent problem-formulation with standardized inputs, general geometric methods, and optimization techniques at the three fundamental scales and the process of unification must ensure compatible methods and data structures.

Semi-automation has been discussed in space planning by Liggett(2000) where the optimization process is dynamic to respond to user interaction. While any optimization process will require several iterations, the updates to the scheme can be devised to allow the user to interact while the scheme is updated. This feature provides scope for development across input interface, constraint generation, and optimization.

2.10 The Evolution of Hypotheses

2.10.1 A common class of linked design problems

The prior research demonstrates the use of topological representation (sections 2.3 and 2.8) to develop floor plans, street network optimization, and analysis of urban environments. This representation is an intermediate model of the physical forms of spaces and activities in a building layout or streets and building locations in an urban layout of city-blocks, etc. The equivalent topological models of the built environment i.e. a graph with nodes and vertices makes it possible to formulate SAP with common methods across the scales to develop an elegant solution to the topological model. The relevance of mapping physical forms to an abstract algorithmic form has been noted separately by for space planning (eg. Eastman, 1973), urban planning (Lynch, 1982), and network analysis (Hillier and Hanson, 1988).

2.10.2 Patterns of geometry and objectives

While the topological models of SAP in design problems are common, the differences in the physical representation are addressed by formulating general geometric operations that can generate process-specific solutions from the optimized topological models. Apart from Alexander et. al. (1977), surveys conducted by Steadman (2010, 2014), the commercial success of rule-based and graph-grammar systems such as City Engine by Parish and Muller (2001), and other prior research presented in sections 2.4 and 2.5 lead to the hypothesis that geometric forms in architectural layouts follow patterns which can be approximated by geometric operations.

While space planning is addressed by adjacency relations or orientation, the scaling of space-activity relations increases the complexity of the graph. At the urban-scale, groups of activities are analyzed using gravity-index, centrality, betweenness, etc, which are nuanced versions of the separation and attraction. The similarities in problem formulation, and constraints observed, imply that the problems can share methods and data structures or they can be connected across scales.

2.10.3 Explicit Design Process

The structuring of design knowledge has been an implicit aim in computational design where the accumulation of rules or patterns and their concatenation is theoretically applicable to design problems. March (1976) proposed the PDI or production, deduction,

and induction reasoning to synthesize a solution from a catalog of elements, predict the performance of the solution and assimilate elements into the catalog. The PDI model, based on Pierce's hypothetical-deductive reasoning has informed reasoning in the general design of objects and systems (Roozenburg, 1993; Johnsaon-Laird, 2006; Reichertz, 2010, and Cramer-Peterson et. al., 2015). The hypothetical PDI loop continuously generates solutions to design problems with an increase in *types* of solutions when it is used repeatedly and applied to different design projects.

While the PDI-model is hypothetical, a computational framework of interconnected SAP-models and the development of features for semi-automation are steps in the realization of the PDI model. The models serve as decision-making artifacts, which can be saved, parsed, and redeveloped over several projects. They can be reviewed by designers as high-level functions without requirements of additional expertise in computer algorithms.

CHAPTER 3. SPACE ALLOCATION TECHNIQUES (SAT)

The space allocation techniques (SAT) described in prior research informed the development of the hypotheses to generalize the SAP models across scales and topologically variant design problems. In this chapter, the proposed techniques are discussed in detail. The problem formulation and application of general techniques are based on hypotheses (section 3.1). Reinforcement learning techniques are used in this research to optimize the layout (section 3.2). The implementation of reinforcement learning to SAP is explained with applications (section 3.3). The generalization of geometric forms is addressed by encoding commonly used patterns of layout configurations into procedural operations controlled by the optimization modules (section 3.4). While these techniques provide solutions to the typical SAP pertinent to a floor plan design, the graph algorithms for networks of space-activity relations provide solutions to commonly used analytical fields (section 3.5). SAT is bundled into models that organize the flow of information across the algorithms to support a variety of computable tasks in design processes and facilitate user-friendly access. These models have been discussed in the next chapter.

3.1 Overview of Techniques Based on Hypotheses

The hypotheses are concerned with the problem formulation and development techniques to address generative possibilities in architecture and planning. They were used

to develop a set of algorithms, organize the data structures and architecture of the proposed computational framework.

3.1.1 A common class of linked design problems:

Layouts of spaces and circulation are transformed into a topological model for the processing of constraints. This representation operates across scales and allows the development of common modules that are not affected by geometric attributes such as shape, governing constraints, or classification based on design problems. This feature allows the application of the same model (graph or cell array) to a wide variety of SAP in design processes and reduces the complexity of the problems because fewer models are required to address the diverse spatial output of design processes.

Since the models share common data structures and methods, they can be connected to develop a large design problem. As a computational framework, this implies that the output of a process can be applied as input to a subsequent process. The interconnected modules allow designers to develop various permutations using the proposed models.

3.1.2 Patterns of geometry and objectives:

Based on well-known patterns or typologies, specific sets of geometric operations are devised to develop the spatial layout. These operations are embedded in several models

and applied across scales and topologically variant design problems. For instance, (a) subdivision of the floor in space planning is used for site parcellation and (b) a peripheral band of spaces for diverse shapes of floor plate boundaries can be generated by a single module. The generalization of geometric operations is based on standard computational geometry algorithms and curve processing algorithms.

The optimization algorithms are standardized with specific input and output that allows it to operate on various types of geometric operations using their topological state. The output generated by the optimization process is parsed by the geometry modules to locate the spaces and generate their geometric forms based on input attributes. Despite the scaling, interactive networks of spaces are solved using a similar set of algorithms applied to topological models (graph) of the problem.

3.1.3 Explicit Design Process

The explicit design process, in this thesis, does not inform the development of techniques. The utility of the explicit design process is expressed when the SAP is developed using the models which are discussed in the subsequent chapters.

3.2 Topological Optimization using Reinforcement Learning

In this research, the allocation of spaces to activity is controlled by reinforcement learning techniques (Sutton and Barto, 1998). They simulate the process using an agent-

environment interaction. This is used to approximate design processes because design projects are not identical which preclude the existence of voluminous data required to determine correlations between design variables and determine the configuration of spaces. Reinforcement learning simulates the design process using the proposed models and generates the data required to find the optimal solution.

3.2.1 Topological Optimization using Reinforcement Learning

Topological requirements in space allocation problems include proximity relations, circulation, orientation, obstructions to a route, etc. The orientations include north, south, east, west, north-east, north-west, south-east and south-west. These relations are sufficient to generate an appropriate layout at an initial stage of design. An overall configuration of spaces is evaluated based on numerical inputs provided by the user.

The topological requirements are satisfied by ***sequential decision-making***. It uses a model-free prediction to determine the optimal action-value function of SAP. Key elements of the method are:

i. The ***environment*** is the set of all possible locations of each space. It contains information about rewards and constraints. The locations are internally represented as ***states*** ($s_i \in S$) and used as arrays. The probability of moving space from one location to another is the ***transition matrix***. This is the probability of reaching the next state **s**'∈S from '***s***' for action '***a***'. For a Markov Decision Process (MDP), a

transition matrix has Markov property P(s'|s, a), where P(s'|(s_n,a_n), (s_{n-1},a_{n-1}), (s_{n-2},a_{n-2})….(s_1,a_1))= P(s'|s_n,a_n)

ii. An ***agent*** interacts with the environment to determine an appropriate location for spaces based on a numerical reward or feedback from the environment. This is implemented as a sub-routine which associates a space with a location.

iii. The ***action*** ($a_i \in A$) taken by an agent is the allocation of a space to a state.

iv. A ***reward*** ($r_i \in R$) is the numerical value provided by the environment to the agent when it takes an action(a) in a state(s). The sole purpose of the agent is to maximize rewards. It is computed from numerical values provided for adjacency and access to cardinal directions. A **discounted reward** coefficient γ (where $0<\gamma<1$) is used to compute the consequences of actions and states explored by the agent in incremental time steps.

v. The ***value function*** v(s) is used to estimate the cumulative rewards that an agent can accrue over a large number of actions. The ***action-value function*** q(s, a) provides the cumulative rewards based on state and action.

vi. ***Policy*** π(a|s) is a map of an agent's action at a state. It is determined by computing value functions for all states. An optimal policy π_*(a|s) has a maximum value function.

vii. ***Learning*** in space allocation involves model-free methods where the agent predicts a value function for the policy and subsequently reduces the error by feedback received from the environment as it explores states which maximize the value

functions. An optimal policy is achieved by an agent-environment interaction that provides feedback to update action-value functions.

3.2.2 Numerical Rewards or Feedback Signal

The space allocation problem is *NP-complete* (section 2) but if a candidate solution is provided, adjacency and directions of the spatial objects can be *evaluated in polynomial time* $O(n^2)$ and $O(n)$ respectively. The *action* used for space allocation is the relocation of a spatial object which is represented by associating the index of a spatial entity with the index of permissible locations. The effect of each action can be measured by a feedback signal in the form of a numerical reward. The numerical reward can be positive to signify favorable formations or negative to indicate a change in the actions. The reward is computed using the user-inputs for proximity relations and access to the desired orientation. Two primary measures are calculated for the action on space and the effect on the entire layout. These are:

i. The score for adjacency relationships between spatial objects is given by equation (1) and the effect on the generated layout is given by equation (2).
ii. The score given by the orientation of space is given by equation (3) and the effect on the layout is evaluated by equation (4).

$$a = \sum v(a,n) / d(a,n),\ \forall a,n \in N,\ d(n,m)>0 \quad (1)$$

$$E_1 = \sum v(n,m) / d(n,m),\ \forall n,m \in N,\ d(n,m) > 0 \quad (2)$$

$$o = \sum v(a,n).x.d,\ \forall n \in N,\ x \in X,\ d \in D \quad (3)$$

$$E_2 = \sum v(n,m).x.d,\ \forall n \in N,\ m \in D,\ x \in X \text{ where} \quad (4)$$

Where,

- N is the set of spaces
- $v(x,y)$ is the adjacency value between x and y
- $d(x,y)$ is the distance between spaces x and y
- D is the set of direction constraints
- X is a set of coefficients {0, 1} that depend on D and calculated by equation (5)

$$\text{If } \forall n \in N,\ d \in D\ \Lambda\ d(n) \neg 0, \text{ then } x_i \in X : x_i = 1 \text{ else } x_i \in X : x_i = 0 \quad (5)$$

3.2.3 Learning from interaction

The optimization module is used to find a location for each space without generating the geometry. It operates on the topological model of the problem. From the geometry modules, an array of locations and spaces are provided. The array of indices of the location array form states. A layout can be represented by mapping the indices of the space array to the location array. This representation is exploited by the agent which takes action to assign spatial objects to appropriate locations. The value of an action is computed

from the user-inputs for proximity relations and direction. This is the feedback signal or reward function which guides the agent.

A space allocation problem is represented as a Markov Decision Process (MDP). In typical layouts, the MDP (equation 6) has a large number of states and actions. The appropriate location of space is found using a model-free approach where the transition probability matrix and rewards are not required. This is an efficient learning method that uses a greedy algorithm to map spaces to states (location) that maximizes the action-value functions of a policy. The value function for a policy is predicted along with an error function for the states in that policy. The agent-environment interaction is the exploration of states which update the predicted value function based on the feedback provided by the environment or learns the predicted value. The agent learns an optimal behavior for the policy by minimizing the error and converging towards the true value of the states. An optimal policy maximizes the value function or action-value functions. A Markov Decision Process (MDP) that represents the environment is defined as

$$L= \{S, A, P(s'|s,a), R(s',s), \gamma\} \quad (6)$$

where $\boldsymbol{S}$ is the set of states, $\boldsymbol{A}$ is the finite number of actions, $\boldsymbol{P}$ is the state transition probability matrix, $\boldsymbol{R}$ is the reward function and γ is the discounted reward.

A policy π is defined as a distribution of actions for each state. A policy can be used to describe the behavior of the agent at a state(s) taking an action(a) at time step(t). In the case of a layout, the policy π is the association of activity with a place in the layout.

$$\pi(a|s) = P[A_t = a \mid S_t=s] \tag{7}$$

The policy is stationary when there is no action which results in a positive value:

$$A_t \sim \pi(\bullet|S_t), \forall t > 0 \tag{8}$$

Thus the discounted reward for a policy is given by calculating the sequence of actions:

$$G_t=R_{t+1}+ \gamma R_{t+2} + \gamma^2 R_{t+2} +... = \Sigma \gamma^k R_{t+k+1} \tag{9}$$

Using the definition of a policy, the state-value and action-value functions are defined. The state-value function $v_\pi(s)$ gives the long term expected rewards of state(s). Action value function $q_\pi(s,a)$ is the expected return starting from state(s), taking action(a) and following policy(π). Bellman's expectation equation decomposes the two functions into immediate reward and expected rewards (bootstrapping). Typical forms of v(s) and q(s, a) for a policy π:

$$v_\pi(s) = \Sigma_{a\in A}\pi(a|S) (R^a + \gamma\Sigma_{s'}P_{ss'}v_\pi(s')) \tag{10}$$

$$q_\pi(s,a) = R_s + \gamma\Sigma_{s'}P_{ss'}\Sigma_{a\in A}\pi(a'|s') q_\pi(s',a') \tag{11}$$

An optimal policy $\pi \geq \pi'$ if $v_\pi(s) \geq v_{\pi'}(s')$ $\forall s$. The optimal policy is found by maximizing value (v^*) and action-value functions (q^*), such that optimal variables are:

$$v^*(s) = \max_\pi(v_\pi(s)) \text{ with complexity of } O(mn^2) \tag{12}$$

$$q^*(s,a) = \max_\pi(q_\pi(s,a)) \text{ with complexity } O(m^2n^2) \quad (13)$$

An optimal policy π' can be found by an ∈-greedy improvement concerning candidate policy π using

$$q_\pi(s, \pi'(s)) = \Sigma_{a\in A}\pi(a|S)\, q_\pi(s,a) \quad (14)$$

This action-value function is updated by greedy explorations of states using:

$$Q(S, A) := Q(S, A) + \alpha(R + \gamma \max Q(S', A') - Q(S,A)) \quad (15)$$

Since a greedy algorithm is used, continual exploration is required to ensure all states are accounted for. This is a simple epsilon-greedy algorithm where (1-ϵ) chooses greedy behavior and ϵ results in random action.

The expected reward of a value function at state(s) under policy π is not directly computed but predicted (*bootstrapping*). Over many iterations, this prediction is updated and converges to the true value. The simplest update $v(s_t)$ towards $R_{t+1}+ \gamma V(s_{t+1})$ is given by:

$$v(s_t) := v(s_t) + \alpha(R_{t+1} + \gamma v(s_{t+1}) - v(s_t)) \quad (16)$$

where

- $R_{t+1} + \gamma v(s_{t+1})$ is the target
- $\delta_t = R_{t+1} + \gamma v(s_{t+1}) - v(s_t)$ is the error

- And $v(s_t) := v(s_t) + \alpha\delta_t$

The true unbiased value of a policy π is $v_\pi(s_t) = R_{t+1} + \gamma v_\pi(s_{t+1})$ and $R_{t+1}+\gamma v(s_{t+1})$ used in the algorithm is a biased estimate which must be updated. The algorithm converges to the maximum likelihood Markov model solution to MDP <S, A, P, R, γ>.

3.2.4 Scaling the learning algorithm

Reinforcement learning algorithms can use supervised learning to predict estimated rewards and update the error function by stochastic gradient descent using simulated data during the exploration of states. Exploring a policy is the reduction of the error function and maximizing the value function. Using methods seen above, simulated data:

$$D=\{ <s_1, v_1^\pi>, <s_2, v_2^\pi>, <s_3, v_3^\pi>, \ldots ,<s_T, v_T^\pi> \} \quad (17)$$

The reinforcement learning module can be scaled with a function approximator such as a linear combination of weights, neural networks, decision trees, etc. The value and action-value functions are modified as:

$$\boldsymbol{v}(s,\boldsymbol{w}) \approx v_\pi(s) \quad (18)$$

$$\boldsymbol{q}(s,a,\boldsymbol{w}) \approx q_\pi(s,a) \quad (19)$$

where $\boldsymbol{w}$ is the function approximation.

In neural networks, the MSE (mean-squared error) between approximate action-value functions q(S,A,w) and true action-value function $q_\pi(s,a)$. This is the minima of a function J(w) where:

$$J(w) = E_\pi[(q_\pi(S,A) - \boldsymbol{q}(s,a,\boldsymbol{w}))^2] \quad (20)$$

The minimum is found using stochastic gradient descent which updates Δw:

$$\Delta w = -\tfrac{1}{2}\,\alpha \nabla_w J(w) = \alpha(q_\pi(S,A) - \boldsymbol{q}(s,a,\boldsymbol{w}))\nabla_w \boldsymbol{q}(S,A,W) \quad (21)$$

There are practical problems associated with this technique which was resolved using known methods (Scaul, et al, 2016). Since the expected value of $q_\pi(S, A)$ changes with every exploration, the neural network will fluctuate. This can be stabilized by using a fixed old value which is updated after α iterations. In effect, two networks are maintained. When states being explored are sequential, the expected rewards are correlated. Independent Identical data is generated using the e-greedy approach described above. The algorithm used to solve topological constraints in space allocation is a Deep Q-Network or *DQN*:

i. Take action according to a ϵ-greedy policy.
ii. Simulated data is the transition stored as $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory D.
iii. Random distribution is sampled from a mini-batch of transitions (s, a,r,s') in D
iv. Compute Q-learning targets from old fixed parameters *w*
v. Optimize the *mean squared error* between Q-Network & Q-learning targets.

3.2.5 Algorithms to solve MDPs

There are three primary algorithms to solve the MDP:

i. *Value Iteration*: $V^*(s)=\max[R(s,a) + \gamma\Sigma\, p(s' \mid s,a)\, V^*(s')]$ using the algorithm:

   For all states (s) :

      For all actions (a):

         $Q(s,a) \leftarrow -R(s,a)\ \gamma\Sigma p(s'|s,a)V^*(s')$

   $V(s) \leftarrow \max(Q(s,a)$

   Until V(s) converges, where, convergence is given by: $|V(t+1)-V(t)|<\epsilon$

ii. *Policy iteration*: $\boldsymbol{\Pi}^*(s)=\max[R(s,a)+\gamma\Sigma p(s'|s,a)V^{\boldsymbol{\Pi}}(s')]$

   Initialize **Π**' arbitrarily

   Loop:

      Compute V(s) for **Π**(s) policy: $V_{\boldsymbol{\Pi}}= R(s,\boldsymbol{\Pi}(s))+\gamma\Sigma p(s'|\boldsymbol{\Pi}(s,a))\ V_{\boldsymbol{\Pi}}(s')]$

      Improve policy at each state: $\boldsymbol{\Pi}'(s)=\text{argmax}([R(s,a)+\gamma\Sigma p(s'|\boldsymbol{\Pi}(s,a))\ V_{\boldsymbol{\Pi}}(s')]$

      Until convergence: $|\boldsymbol{\Pi}(s)-\boldsymbol{\Pi}'(s)|<\epsilon$

iii. *Q-learning*: $Q^*(s,a)=\max[R(s,a)+\gamma\Sigma p(s'|s,a)V^{\boldsymbol{\Pi}}(s')]$

   Initialize discounted factor γ, learning rate $\propto$, transition matrix R+Q(s,a)=0-matrix.

   For each episode(agent goes from starting state to final state):

   Select a random initial state

Do while(the terminal state is not reached):

Select action ***a*** for current state ***s***

Use this action ***a*** to go to next state ***s'***

Calculate max Q(s',a') value for next state s'

$Q(s',s)=Q(s,a)+\propto[R(s,a)+\gamma \max[(Q(s',\text{all actions}) - Q(s,a)]]$

Set next state as s' and current state=s, s'=s

End do

End episode

Among these standard methods, Q-learning has been used with dynamic programming and McMC to generate the transition matrix. The equation to maximize Q-value is given by:

$$Q(s,a)=(1-\propto)Q(s,a)+\propto[R(s,a)+\gamma \max(Q(s',a)] \quad (22)$$

3.2.6 Function Approximator using neural networks

The data representation (Chollet F, 2017, Goodfellow I, 2016) is in the form of tensors that have attributes of axes (rank), shape, and data-type. The axes are dimensions of the data being input, for instance, a vector is a 1-d tensor, and the matrix is a 2-d tensor. The shape of a tensor is the number of dimensions in each axis. The data type can be integer, float, boolean, etc. Examples include time-series data of 2d or 3d, images with 4d tensors for samples, width, height, channels, or video which have 5d samples, frames, width, height, channels. For most common uses, data is in vector format. Commonly used

tensor operations are dot products, element-wise operations (activation), broadcasting, and reshaping. To find the pattern using backpropagation, the most well-known algorithm is stochastic gradient descent, which slightly alters the weights of the graph until the error is minimized. It is a gradient optimization where chain-differentiation is used to find the error propagated by weight in the network. The four elements for neural network design are:

1) Layers of nodes, edges that form the network
2) Input data from which errors can be determined
3) The loss function is the feedback signal for backpropagation
4) Optimizer to determine the learning method

Deep learning ie neural networks with many hidden layers is a computational model. Although they were initially inspired by the human brain, they are best understood as mathematical models. They are typically modeled as directed acyclic graphs (DAG) and the learning mechanism of a neural network relies on feedforward and backpropagation.

Initially, the neural network is initialized by random weights. Feedforward is the propagation of the inputs (I) with a matrix of initial weights (X). It is calculated by equation (22). The activation function used is a sigmoid function given by equation (23). The Mean Squared Error (MSE) is found by equation (24), where t_k is the target, and o_k is the output. The error contributed by an edge is given by equation (25). Thus the new weights are given by equation (26), where α is the learning rate, $(0< \alpha <1)$.

$$O=S(dot(I,X)) \tag{22}$$

$$S(x)=1/1+e^{-x} \quad (23)$$

$$E=(t_k - o_k)^2 \quad (24)$$

$$\partial E/\partial(w_{jk})=-(t_k-o_k)S(\Sigma w_{jk}o_k)(1-S(\Sigma w_{jk}o_k)).o_j \quad (25)$$

$$\triangle w_{jk} = \alpha E_k o_k (1-o_k).o_j^T \quad (26)$$

Backpropagation Calculations:

i. New weights = old weights + learning rate x error propagated

ii. Using the chain rule, change in weights is given by

$$\partial E/\partial(w_{jk})=\partial(E)/\partial(o_j)x\partial(o_j)/\partial(\Sigma w_{jk}o_j)x\partial(\Sigma w_{jk}o_j)/\partial(w_{jk}) \quad (27)$$

iii. Calculate for 1,2,3 nodes of output layer wrt 1,2,3 nodes for hidden layer and substitute corresponding values shown in the graph and calculations.

3.3 Mapping Reinforcement Learning to Space Allocation Problems

This section describes three applications of reinforcement learning algorithms. The applications include pathfinding for circulation, placement of spaces in a plane, and organization of cells of a given layout based on adjacencies and orientation.

3.3.1 Pathfinding

Routing is a common problem studied in artificial intelligence and space allocation. It is a search for the most efficient route through a set of cells. An algorithm or the agent must navigate the cells while maximizing the rewards. From a cell, it can move to any permissible cells. The choices lead to a combinatorial explosion. Numerous restrictions are placed concerning the cost of traversal between cells. The agent determines the most favorable next cell at each cell. An efficient algorithm is demonstrated by finding a path through the cells with minimum steps. The algorithm is considered scalable if it can operate on an increasingly larger number of cells or states with similar time complexity.

The NP-completeness (Jo and Gero, 1998) of SAP is addressed by such scalable algorithms by reducing the problems to SAP. The SAP is represented as a graph G(V, E) where the nodes represent the cells. The agent assigns spaces to nodes as it traverses the graph. An optimal layout is determined when the arrangement of spaces that resolve the constraints. This strategy for optimization is applied to space planning, site feasibility, and space-activity networks.

This research uses SARSA, policy, and Q-learning algorithms (section 3.1.5). Each cell is a state. The cells are represented as an array. The agent takes action by choosing the next cell or simply querying the elements of an array. If the agent reaches the terminal state, it receives a reward. Or, the sequence of states in an iteration is scored (equation 13). The agent ranks the states encountered using the discounted rewards (R) with equation-15. Over several iterations, the agent can determine the most favorable path at each cell or state

(Table 4) illustrated by Figure 2. This is a process of sequential decision making where the agent-environment interaction is used to solve a difficult problem.

Overall Learning Strategy:

i. Initialize Q matrix with elements set at 0

ii. Discounted rewards Reward: $R(s,a) = R(s,a)+\sum_{(i=1)}^{T}\gamma^{(i-1)}R_i$ (where $0<\gamma<1$, let $\gamma=0.8$)

iii. After m x n iterations Q = Scoring & Policy: best move for each state

iv. Set of states s_i: {(0,0, (0,1), (0,2), …, (6,5), (6,6)}

v. Set of actions a_i: {up, down, left, right}

vi. Starting state: {A}

vii. End State: {I}

Table 4. Transition matrix based on the computations

States(x,y)	1	2	3	4	5	6
0	1.80	2.25	2.81	3.52	2.81	1.80
1	0.00	0.00	0.00	4.40	0.00	0.00
2	10.74	8.59	6.89	5.50	4.40	0.00
3	13.42	0.00	0.00	0.00	0.00	0.00
4	16.78	20.79	20.21	32.71	0.00	0.00
5	13.42	0.00	0.00	40.96	0.00	0.00
6	10.74	8.59	0.00	51.20	0.00	0.00

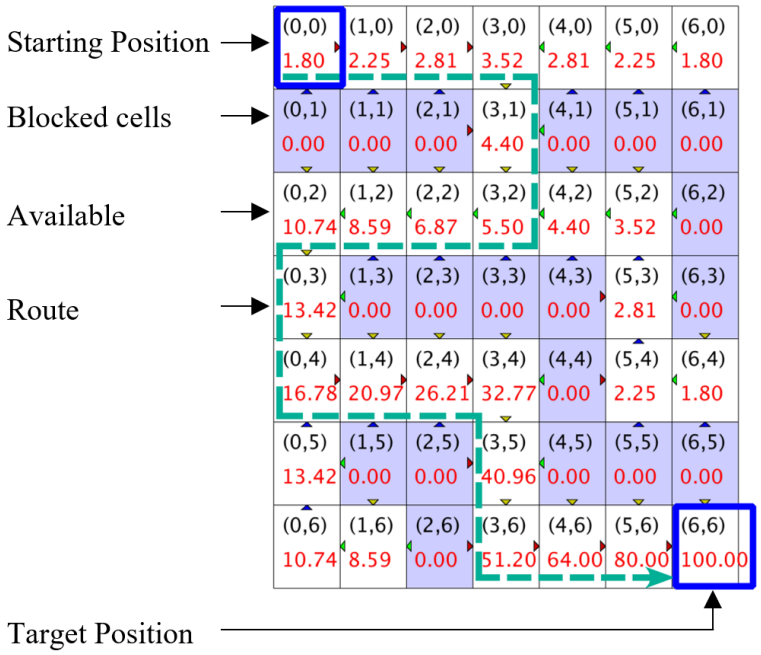


Figure 2. Routing/Pathfinding is a typical problem in Q-learning

3.3.2 Accretion of spaces

In this section, the agent-environment interaction is used to demonstrate a combinatorics problem that generates floor plates and allocates activities by accretion of spaces using inputs of area requirements and proximity relations between them (Table 5). A variation of this problem is used in site feasibility studies and generating hierarchies of a street network.

In this problem, the spaces branch out from a point. Apart from geometric and topological constraints, the objects may be additionally bound by requirements such as regional properties, orientation, the proportion of space occupied. Markov chain Monte

Carlo simulation is used to solve this problem. Although, spatial requirements can be plotted with relative ease. Matching geometric forms with adjacency relations is a non-trivial problem arrangement problem. Since the dimensions may vary, they cannot be easily packed into grids or solved by typical bin-packing problems. However, if a candidate solution is provided, adjacency constraints between geometric forms of the spatial objects can be verified easily, indicating that backtracking can be employed efficiently to eliminate sub-trees of a search process and greatly reduce the steps of a search process. The proposed method (Figure 3) generates the floor plate by placing spatial elements based on input dimensions and ensures that the local neighborhood has appropriate adjacencies or displays the missing relations upon completion (Figure 4).

In this process, an architectural layout is generated by assigning geometric form and location to each spatial entity such that proximity relations are maintained. To achieve this, the spatial requirements are organized into a stack and a tree (graph $\boldsymbol{G}$) is constructed from the desired proximity relations. It is the ideal graph of parent-child spatial relations with a value representing the strength of the connection. The first node of the stack is placed at a pre-defined position and its geometry is plotted based on the given dimensions. An iteration is set up to extract nodes from the stack and position them based on the vertices of the existing geometry (Figure 3). Subsequent nodes select an existing spatial entity and query available vertices for placement. The preferred node has a higher adjacency value. The probability of reaching disconnected nodes is low in the early stages of growth because sorting operations are performed to organize the stack. But there is no restriction on choosing a node with no value. Along with the vertex, a vector indicating the direction of

diagonals of the space is required to generate the geometric form. This is selected to provide the best immediate result. Selection criteria prevent overlap between spaces. This is a greedy algorithm that seeks to (objective function) maximize proximity relations and minimize the ratio between overall length and width.

Initial iterations produce an imperfect solution that is rectified in a stepwise manner. A proximity relation between two spaces is established if an edge is shared between the polygons representing their geometry. At the end of a generation, graph ***S (V', E')*** is created by considering common segments between any two polygons. It is the set of any spatial-pair that shares an edge. The parent-child relationship between them is given by the order of spatial elements in the initial stack. The set of such pairs is compared with the ideal graph ***G (V, E).*** For each relation that exists in ***G*** and not in ***S***, the child element is removed from ***S***, and placed back in the stack of nodes ***G*** to be re-positioned until adjacency criteria are satisfied. This process reduces steps by identifying and improving only the sub-graphs of each generation. Parts of the graph are amended in subsequent generations (Figure 4) without affecting the correct positions. An additional constraint is demonstrated using a ratio between the overall length and width (Figure 5 b). It controls the compactness of the overall geometric form.

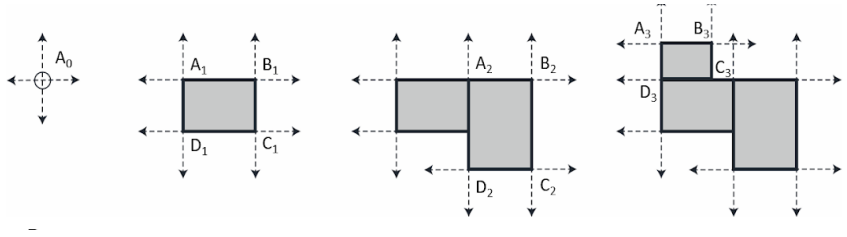


Fig 3 a. Dry Run

(Figure 3 continued)

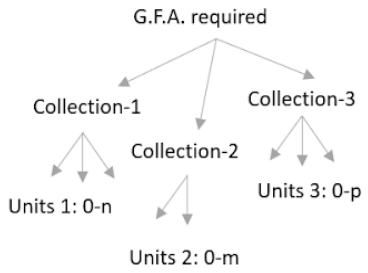


Fig 3 b. Topological Representation

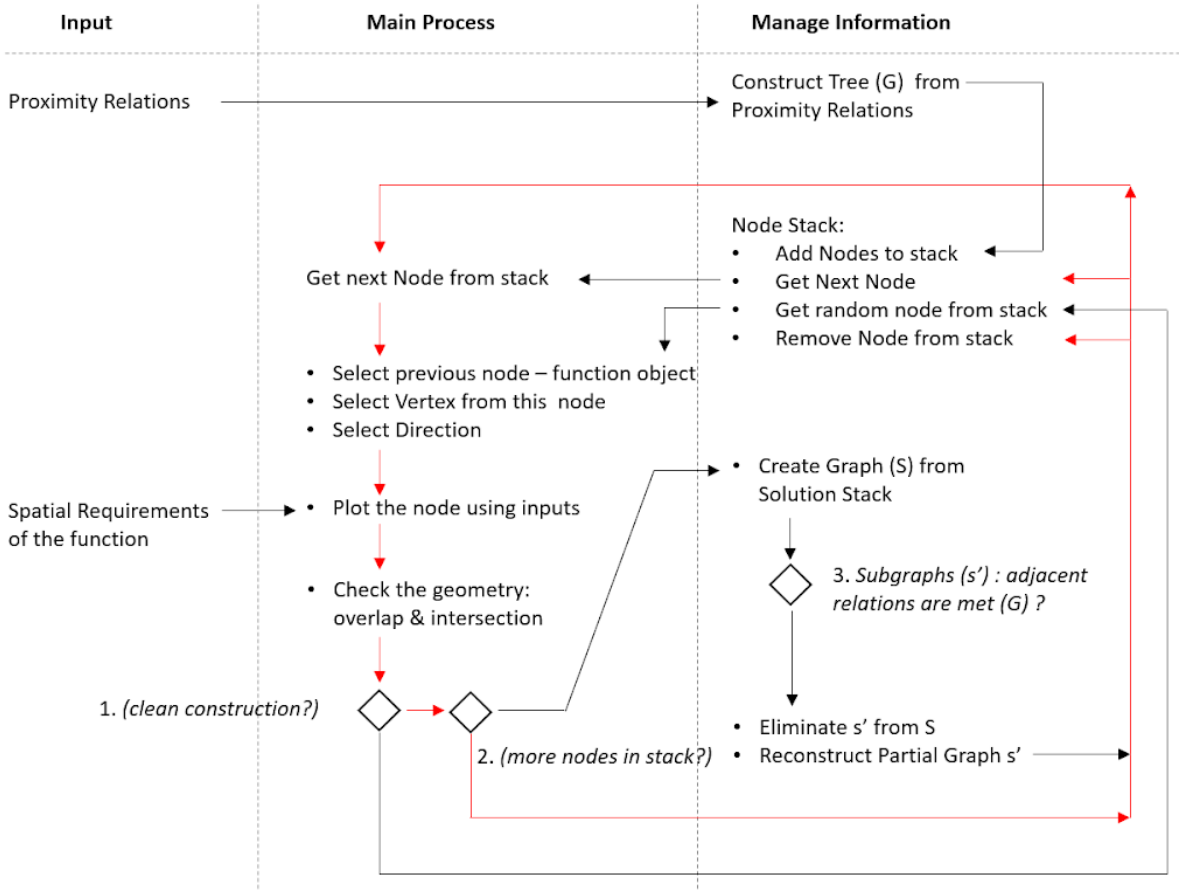


Fig 3 c. Process of Spatial Propagation involving quadrilaterals

Figure 3. Problem set up for spatial exploration

The process is implemented using a table of probability values between the tuple ***T*** given by vertices, nodes, and a direction vector. Initially, the first generation is considered as the ideal spread. Subsequently, when a node is placed, the overall score of that generation is computed. If the action improves the overall score, the table is updated to reflect the probability of ***T***. The increment in probability is given by improvement in score. Eventually, over a few generations, the process converges to maximum possible values for each triplet ***T***. However, a trade-off between proximity relations and minimization of the coverage needs to be addressed. To parameterize the constraint, a numerical range for the ratio between the length and width is exposed to the user. If the range is high, a variety of spatial organization is generated.

The technique for spatial exploration was tested with numerous spatial requirements and proximity relations (Table 6). The number of iterations required was checked by systematically changing the spatial requirements and adjacency relations. The largest set of requirements was composed of 10 spatial entities with 10 relations between them. Multiple runs (1000) generated solutions between 20 and 47 generations.

Table 5. Program Constraints

Name	Length	Width	Red	Green	Blue	F0	F1	F2	F3	F4	F5
F0	100	100	255	0	0		10				
F1	70	70	150	15	0				10		
F2	100	200	0	255	0				10		
F3	300	100	0	255	0						10
F4	300	200	0	0	0						10
F5	200	200	0	150	0						

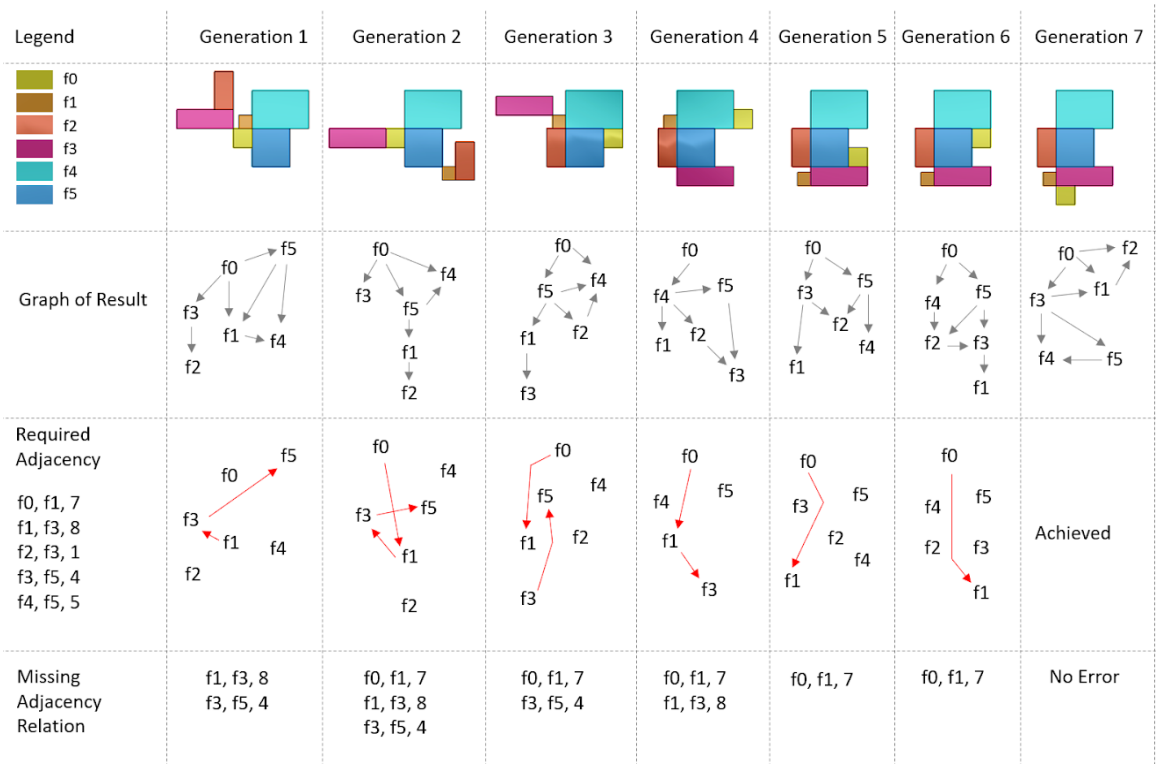


Figure 4. Feedback and backpropagation in spatial exploration

The stochastic optimization technique for spatial exploration generates rational spatial entities (floor plates or location of buildings) from a set of program requirements and their proximity relations. This is referenced against a graph with the best relations. The process estimates an approximation of the ideal graph over several generations by iteratively improving the spatial organization. To demonstrate additional constraints, the overall ratio of the length and width of the region occupied is controlled to restrict excessive spread.

Table 6. Larger Requirements of program and adjacency

Functional (spatial) Requirements

Name	Length	Width	Red	Green	Blue
f0	100	100	255	255	0
f1	70	70	250	150	0
f2	100	200	250	55	0
f3	300	100	255	0	150
f4	300	200	0	255	255
f5	200	200	0	150	255
f6	200	200	0	50	255
f7	225	250	250	0	255
f8	215	275	150	0	255
f9	200	325	50	0	255
f10	175	175	255	0	155

Adjacency Relation

Obj A	Obj B	Value
f0	f1	8
f1	f3	4
f2	f4	4
f3	f5	5
f4	f5	9
f5	f7	1
f6	f8	9
f7	f5	7
f8	F9	9
f10	f9	3

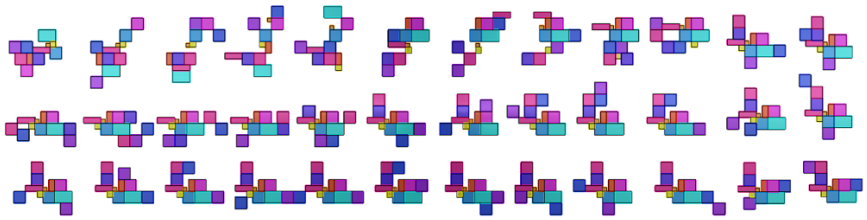


Fig 5 a. Sample of iterative improvements in layout

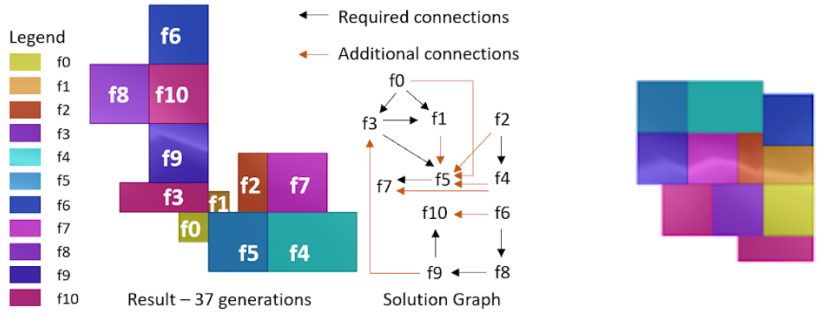


Figure 5 b. Optimal Solution found w/o compactness constraints (left) and with *compactness* constraint (right)

(Figure 5 continued)

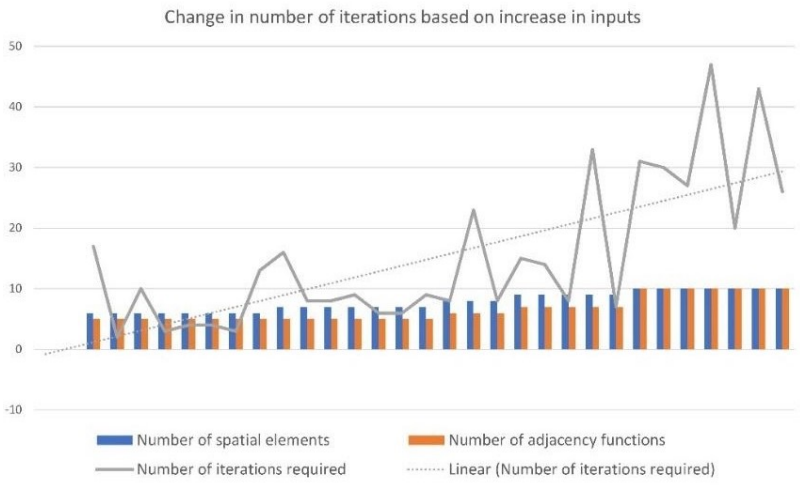


Figure 5 c. Change in number of iterations with an increased input

Figure 5. Spatial exploration on a large input set

3.3.3 Reinforcement Learning in SAP

The application of reinforcement learning to SAP is accomplished by operating on an abstract representation of the layout. All extraneous features are ignored including the effect of geometry. Similar to the problem in 3.3.2, the representation of the layout was constructed with 53 equal cells (Figure 6). Voids were introduced to increase the topological complexity. This also increased the complexity in orientation accessed by each cell. The constraints exert a predetermined influence on the system (section 3.1.2). The orientation-value of each cell is saved in an array. Similarly, the pairwise distance between any two cells is calculated and stored in an array. These values remain constant (Figure 6

a, b). Over the numerous iterations, they can be retrieved by traversing the array. They contribute to the computational complexity by O(n).

Using topological structures of layouts, the locations (L) and spaces (S) are treated as a 1-d array (equations 28, 29). Their association (R) is studied in a 2d-matrix (equation 30). The score or Bellman discounted rewards are calculated as usual using the numerical rewards from section 3.2.2. A configuration is reconstructed from an array of space identifiers (S).

$$\text{Locations } L = \{0, 1, 2, 3, \ldots, 52\} \tag{28}$$

$$\text{Spaces } S = \{a, b, c, \ldots, g\} \tag{29}$$

$$\text{Association } R = L \otimes S \tag{30}$$

The problem of space allocation is reduced to associating (equation 30) the two arrays, querying the constraints and evaluating the effect of the association. In this problem, the states are locations (L). An agent takes action by placing a space (S_i) at a state (L_j). Contrary to the prior problem (section 3.3.1), every action receives a reward based on adjacency and orientation (section 3.2.2). Only the actions that result in a favorable reward are accepted.

The policy-iteration algorithm (section 3.2.5) was found to be more efficient than the Q-learning algorithm for layouts. In this case, a random association R' (equation 30) is assumed and iteratively improved. The cells of the layout were constrained by the user

inputs for adjacency and orientation given in Table 7. The string of space identifiers L(a,b,…,g) was correlated with spaces S(0,1,2,…52). This representation allows a layout to be identified by the string arrangement as illustrated in figures 6 and 7.

Table 7. User inputs from which adjacency and orientation constraints are generated

	Adjacency							Orientation/Direction							
Name	a	b	c	d	e	f	g	n	nw	w	sw	s	se	e	ne
A	20	-10	0	0	0	0	0	20	0	0	0	0	0	0	0
B	-10	0	0	0	0	0	0	0	0	0	0	0	0	20	0
C	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	20	0	0	20	0	0	0	0	0	0
E	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	20	0	0	20	0	0	0	20	0	0	0	0
G	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0

Overall Learning Strategy:

1) Initialize Q matrix with elements set at 0
2) Discounted rewards Reward: $R(s,a) = R(s,a)+\sum_{(i=1)}^{T}\gamma^{(i-1)}R_i$
3) After m x n iterations Q = Scoring & Policy: best move for each state
4) Set of states s_i: {(0,0, (0,1), (0,2), …, (6,5), (6,6)}
5) Set of actions a_i: {up, down, left, right}
6) Starting state: {A}
7) End State: {I}
8) Set γ=0.8 (any value where $0< \gamma <1$)

Name = {a,b,c,d,e,f,g}
Location = {0, 1, 2, …, 51, 52}
A cell can have numerous orientations based on the faces. Orientation={north (n), north-west (nw), north-east (ne), south (s), south-west (sw), south-east (se), east (e), west (w)}.

Figure 6 a. Details of a cell of the layout

a (0) n.nw.w.sw.ne.	b (8) n.nw.ne.	c (16) n.nw.ne.	d (22) n.nw.ne.	e (28) n.nw.ne.	e (34) n.nw.ne.	f (40) n.nw.ne.	g (46) n.nw.e.ne.
a (1) nw.w.sw.	b (9)	c (17) s.se.	d (23) sw.s.se.	e (29) sw.s.se.	f (35) sw.s.se.	f (41) sw.s.se.	g (47) sw.s.se.e.ne.
a (2) nw.w.sw.	b (10) e.	Topological Complexity: void space					
a (3) nw.w.sw.	b (11) ne.	c (18) n.ne.	d (24) n.nw.ne.	e (30) n.nw.ne.	f (36) n.nw.ne.	g (42) n.nw.ne.	g (48) n.nw.e.ne.
a (4) nw.w.sw.	b (12)	c (19) s.se.	d (25) sw.s.se.	e (31) sw.s.se.	f (37) sw.s.se.	g (43) sw.s.se.	g (49) sw.s.se.e.ne.
a (5) nw.w.sw.	b (13) e.	Topological Complexity: void space					
a (6) nw.w.sw.	c (14) ne.	c (20) n.ne.	d (26) n.nw.ne.	e (32) n.nw.ne.	f (38) n.nw.ne.	g (44) n.nw.ne.	g (50) n.nw.e.ne.
b (7) nw.w.sw.s.se.	c (15) sw.s.se.	d (21) sw.s.se.	d (27) sw.s.se.	e (33) sw.s.se.	f (39) sw.s.se.	g (45) sw.s.se.	g (51) sw.s.se.e.ne.

Initial Layout is represented as a string:
a,a,a,a,a,a,a,a,b,b,b,b,b,b,b,c,c,c,c,c,c,c,c,d,d,d,d,d,d,d,e,e,e,e,e,e,e,e,f,f,f,f,f,f,f,f,g,g,g,g,g,g,g,g,g,g

Figure 6 b. Initial layout. 1-d array of associated activity and space

Figure 6. Setting up the problem for reinforcement learning

Solution to the problem: The information (data) about the optimization process is extracted to visualize the actions taken by the agent. At each iteration, the score was monitored based on the contribution of the constraint. Figure 8 shows a steady increase in improvements in

the layout based on adjacency and orientation scores until optimality was reached. Figure 9 shows the rate of change of scores, which peaks and flattens to correspond with exploration and exploitation- because the *rate of change* of improvements is not constant. Over several iterations, the agent discovers a better partial allocation (substructure). This period is the flattening of the graph. It peaks at the iteration when the action is committed. In Figure 10, it is seen that a relatively small change in the layout can increase the overall score because each location affects many states. Similarly, in Figure 11, even though the rate of change of score increases and decreases based on the above-mentioned reason, the layout does not fluctuate. This is because string manipulation (change in layout) is constant over time which generates 1000s of values every second and the optimization takes 10-15 seconds (https://IDF-ai.herokuapp.com/).

The sample of constraints extracted from the iterations and used to guide space allocation:

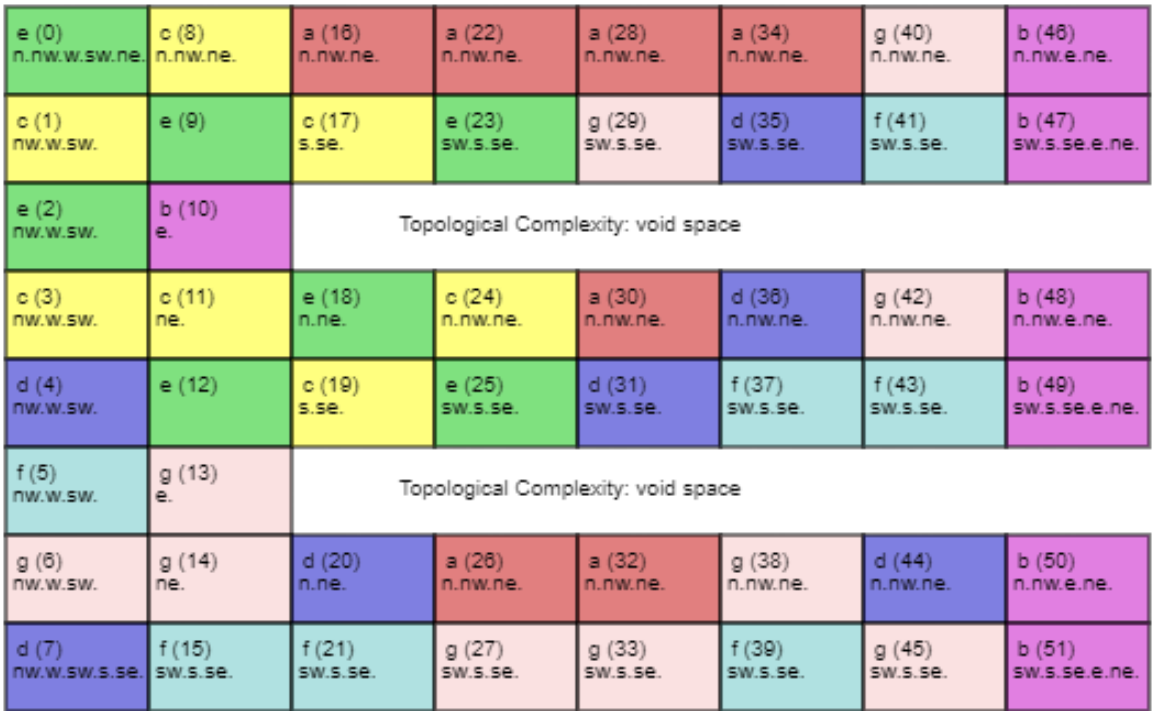
i. Sample of contribution of constraint to the layout (Figure 7):
    a. {constraintId: 1, value to layout: -0.1130466837888443}
    b. {constraintId: 1, value to layout: -0.09701425001453319}
    c. {constraintId: 0, value to layout: -0.07327433054473118},
    d. etc
ii. The adjacency and orientation relations that contributed to the score (sample):
    a. {id: 1, type: "adjacency", detail: "a.b"}
    b. {id: 2, type: "orientation", detail: "a.e"},
    c. etc
iii. The layout string with the effect on adjacency and orientation scores (sample):

a. Total Score: 556.1797885969199

b. Layout Allocation: *a, e, c, d, g, f, g, b, c, e, b, c, e, b, b, d, e, c, e, b, a, f, d, c, c, d, d, d, a, e, a, e, g, g, c, d, a, f, a, f, g, f, g, f, a, g, b, g, b, g, f*

c. Adjacency score: 385.17978859692

d. Orientation score: 175



Iterations: 2180
Best SPACE-ALLOCATION:
e,c,e,c,d,f,g,d,c,e,b,c,e,g,g,f,a,c,e,c,d,f,a,e,c,e,a,g,a,g,a,d,a,g,a,d,d,f,g,f,g,f,g,f,d,g,b,b,b
Best Score: 475.90115403016756,
Best Score Iteration: 3228
Time Elapsed: 13838.15500000492 milliseconds

Figure 7. Optimal layout generated from the inputs.

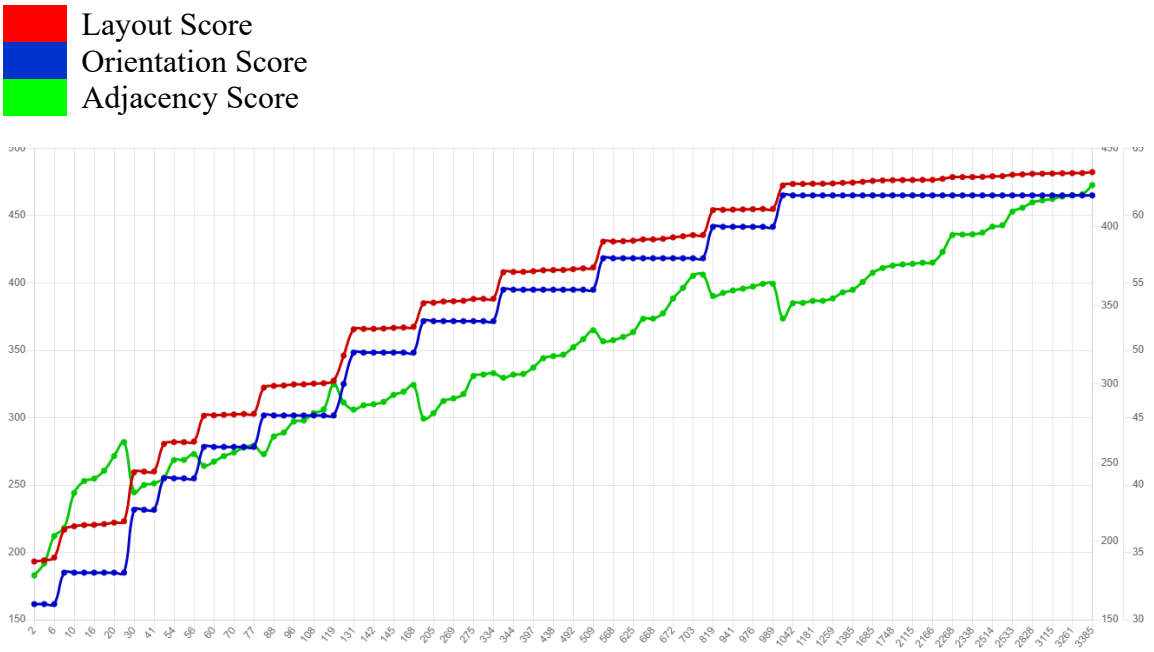


Figure 8. Graph for Comparison of layout-score, orientation-score, and adjacency-score

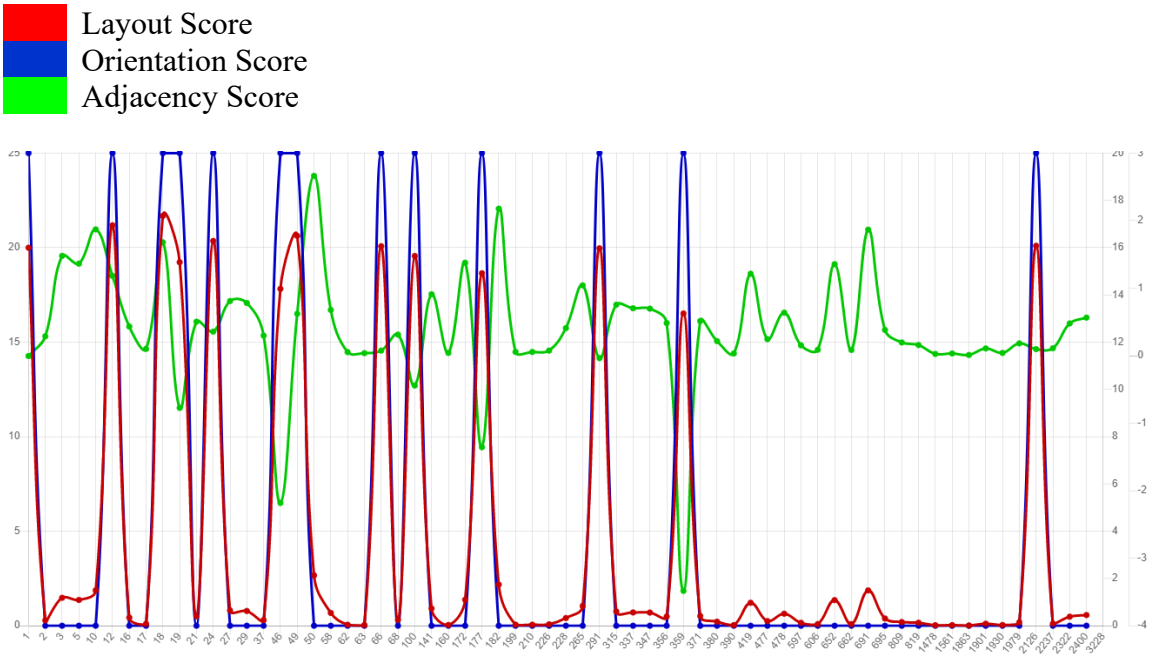


Figure 9. Graph for *Rate of Change* of Layout-Score, Orientation-Score & AdjacencyScore

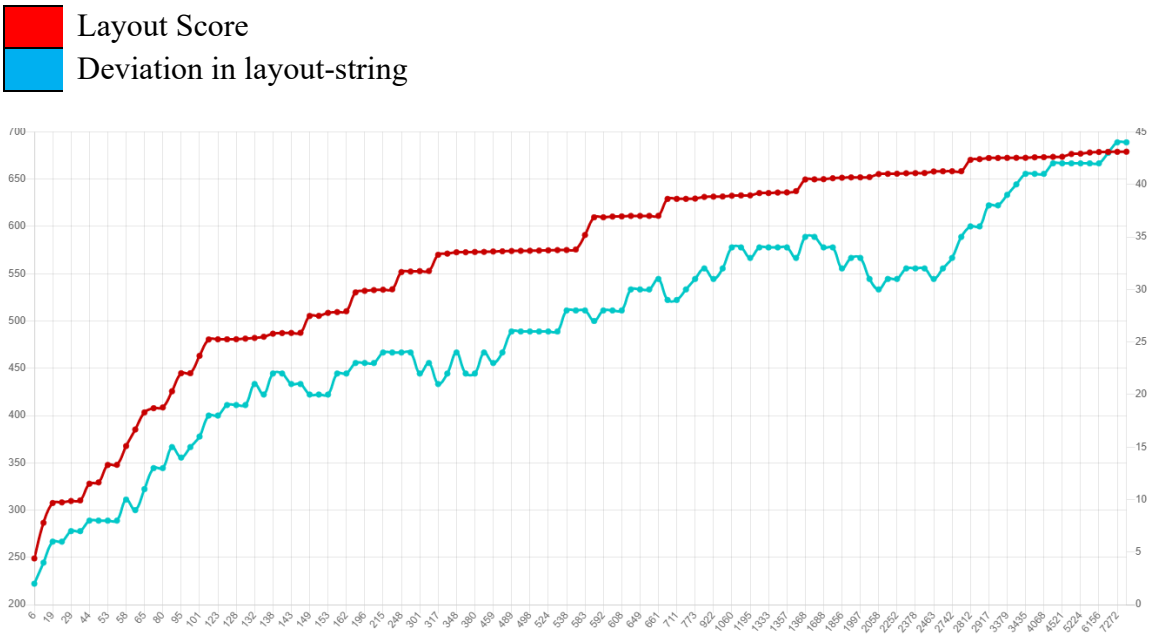


Figure 10. Graph for *Comparison of* Layout-Score, Layout-String

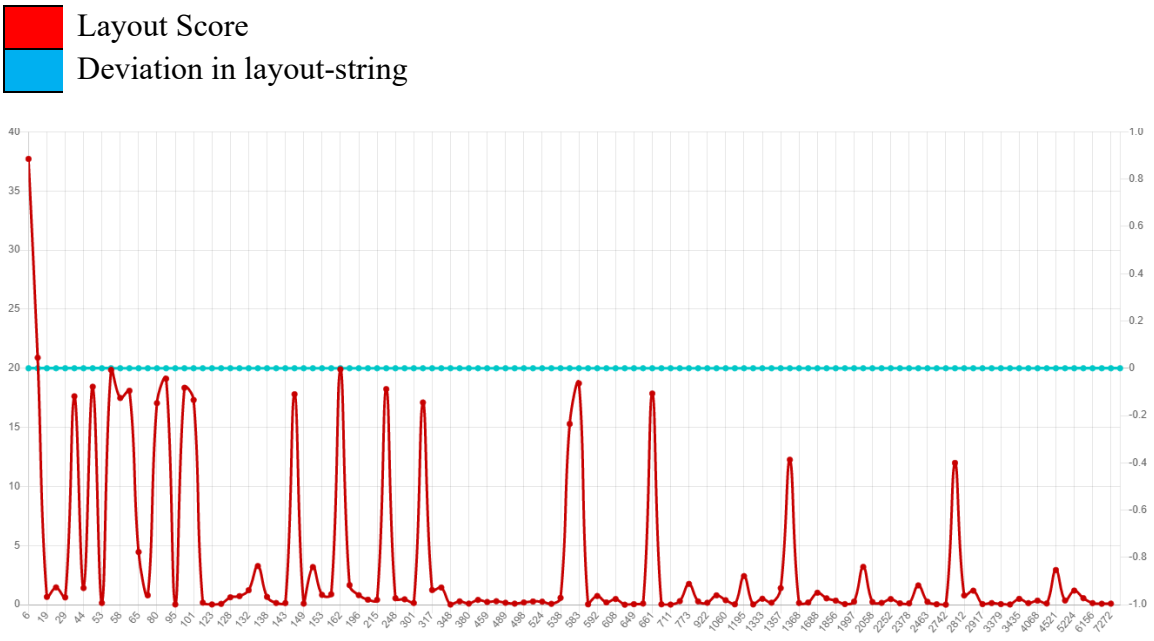


Figure 11. Graph for *Rate Of Change* Of Layout-Score, Layout-String

3.4 Fundamental Geometric Operations

As mentioned, the proposed computational approach is based on the activities organized by constraints, and the appropriate architectural forms generated by geometric operations. The models of SAP are internally optimized using a common reinforcement learning algorithm (section 3.1). The algorithm operates on the topological representation of the problem. It guides the subsequent geometric operations.

The geometric variations in architecture and planning are specific (Figure 25 b). They vary in range from general polygons to curves. They have a predilection for right angles, parallel lines, and arcs. However, such constraints are not necessary and often complex shapes are used. But well-known algorithms for tessellation, space-filling curves, or free-flowing curves are not suitable to generate appropriate architectural shapes. Further, the patterns or rules that are observed in a layout may vary drastically from others. Quite often, the geometry of layouts is generated by composite shapes with orthogonal edges, partial irregularities, or curvature. This results in a non-trivial problem. It requires an elaborate approach based on the algebraic or parametric form of the curve. It must be noted that elaborate *error correction mechanisms* are necessary during the integration with optimization algorithms.

The proposed geometric methods operate on commonly used shapes in design including rectangles, curves, general polygons, open curve segments, and composite sections. They generate appropriate massing and configurations of the layout. To support

various design practices, additional features are incorporated for organizing information between optimization and geometric processes (Table 2).

3.4.1 Geometry of Spaces

Despite the seeming randomness of architectural shapes, certain patterns have persisted. Prior research suggests that the patterns exhibit relations that can be generated by computational processes (section 2). In this thesis, it is proposed that the design solutions are generated by constrained geometric operations. This is computationally elegant and practically feasible when compared to other paradigms of generative systems. This has led to the development of realistic layouts. Based on the underlying algorithms and nature of architectural geometry, three classifications were made:

i. Spaces along a curve: Spaces are generated in a peripheral band along a given curve and bound by normals. The spaces can be placed on either side of an open curve with a circulation passage or spaces along one side of a given closed boundary forming a peripheral band. A peripheral band is recursively extended to fill a region until it meets the termination criteria. This is controlled by the user or automated using constraints such as self-intersection and minimum area or depth of the region.
ii. Subdivisions of a curve: Spaces are generated by recursively subdividing a closed general curve into parts by exploiting a homeomorphic relation between the list of area requirements and geometric operations that partition the curve.

iii. Accretion of spaces: This is a method of accretion where spaces are stacked to satisfy constraints such as adjacency and distance relations. The organization of spaces may be based on a governing logic such as pre-defined template or "concept". For instance, the geometric form of the space is placed on a preceding space based on a feasible point on the boundary.

3.4.2 Spaces Along the Curve

Complex shapes used in the practice of architecture and planning are composed of primitives with partial curves or angled rectilinear components to improve the utilization of spaces. The geometric forms occupy regions under the curve while ensuring that area and adjacency requirements are met. Circulation spines are recursively generated to subdivide the region inside the curve. S*patial entities* are placed along the circulation trajectory and periphery of boundaries.

3.4.2.1 Open Curves

An elementary architectural configuration is the location of spaces along a circulation spine or axis. A region is generated along the axis where the spatial objects can be located. Spaces can be placed on either, or both sides of the spine. Using this construction, spaces can be optimized for adjacency, circulation, and program (Table 8).

Points are generated along the axis (Figure 12) using a parametric equation of the curve. From these points, two normal directions at each point are determined. At each

parameter, points are transformed by a constant bay depth along the normal to generate linear segments (Figure 12 b). The geometry of the space is given by joining the curve segment between subsequent points and segments along the normal. It generates a closed region that matches the area required. In this pattern of architectural design, the bay depths are based on user input and region for circulation are procedurally generated along the spine or after the rows of spatial objects.

Table 8. Program requirements of spaces

Spaces	Name	Area	Depth	Number	Color (rgb)	f0	f1	f2	f3	f4
0	f0	1000	100	2					50	
1	f1	300	100	2			-50			
2	f2	500	100	2				-50		
3	f3	850	100	2					50	
4	f4	350	100	2						100

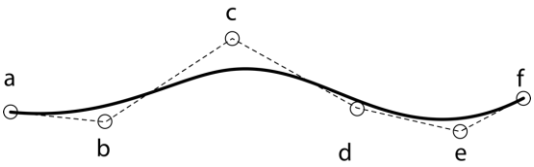


Figure 12 a. Given Input Shape to place spaces

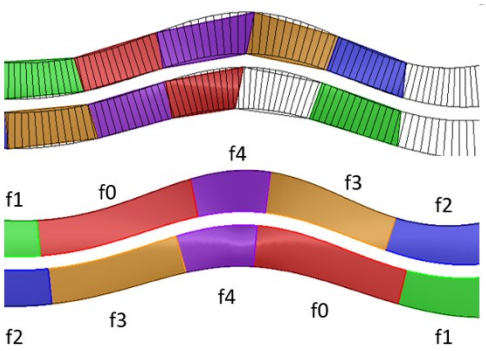


Figure 12 b. Discretization of the initial curve and optimized solution with curvature

Figure 12. Applying the space allocation along an open curve

3.4.2.2 Medial Axis Transform

In this research, it is proposed that composite sections of a given boundary can be computed to generate appropriate spaces based on constraints. Typical configurations of spaces in architecture and planning are determined by segregating the boundary conditions and internal region. This is a method of generating (a) recursive bands of spaces along the periphery, and (b) longitudinal and transverse circulation spines or axes. Using a *thinning interpretation* of the boundary's medial axis, peripheral bands of spaces are recursively generated (Figure 13 a). It determines an appropriate spanning curve across the internal region. It is modified to eliminate the diagonalization at the ends and meet the boundary at a point. This generates a longitudinal axis of circulation from which lateral segments are projected in the two normal directions (Figure 13 b). These are transverse axes of circulation. Spatial entities are located along the axes of circulation (Figure 13 c). Dimensions for circulation, area, and adjacency considerations of spatial units are parameterized as inputs from the user. The layouts illustrated in Figure 13 demonstrates the applications of this construction.

Spatial requirements are embedded in the logic of shape generation. The geometric form of the layout is developed as an elementary vector (a 1-d matrix that indicates direction and magnitude) undergoing transformations and occupying the region. The expression of a shape is a set of connected points. Common error-correcting criteria restrict the movement of the vector. These include the intersection between shapes, orientation, containment within a site, and proportions of shapes. As the vector traces architectural primitives, the geometric attributes are evaluated.

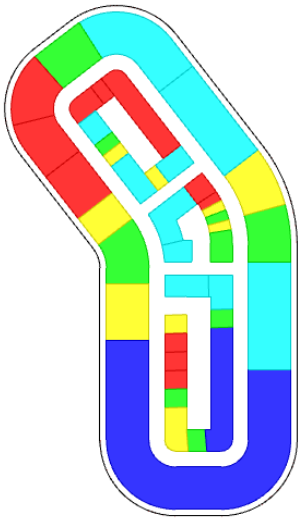


Figure 13 a. Spaces generated by recursively applying peripheral bands of spaces

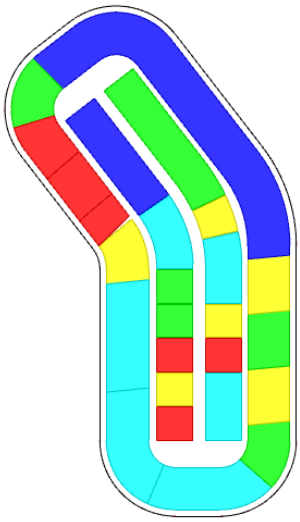


Figure 13 b. Spaces along a longitudinal axis of circulation.

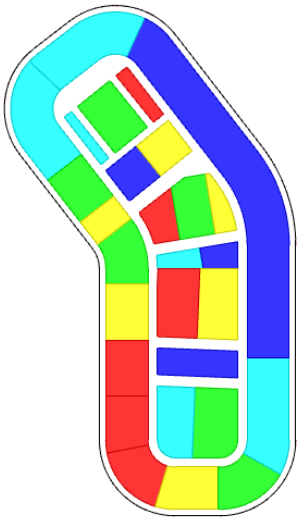


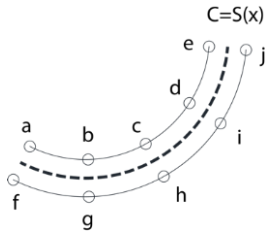
Figure 13 c. Spaces along transverse axes

Figure 13. Various organizations of the spaces along curves

3.4.2.3 Organization: Connection to Optimization Module

To solve for topological optimization, the identifiers are stored in a one-dimensional array or vector with bijective correspondence to the point locations (Figure 14 a). The array of identifiers provide a compact feature vector to accurately represent a configuration of spaces. An alternative layout can be generated by shuffling the indices of the array which reconfigures the relationship between identifiers and point locations. Generating the geometry is a procedural step of incrementally occupying a designated region until area constraints are met. Proximity relations and direction can be calculated

from the geometry which allows a metric measure for relationships based on user inputs (Figure 14 b).



Relations between locations, requirements & constraints:

Array of locations A={a,b,c,d,e,f,g,h,i,j}

Array of spatial requirements L={A,B,C,D,E,F,G,H,I,J}

Array of Constraints (V) = X= U_{ij}^{mn}f($A_i \otimes L_j$) = {$v_1, v_2, \dots, v_k$}

Figure 14 a. Arrays for identification and determining substructures

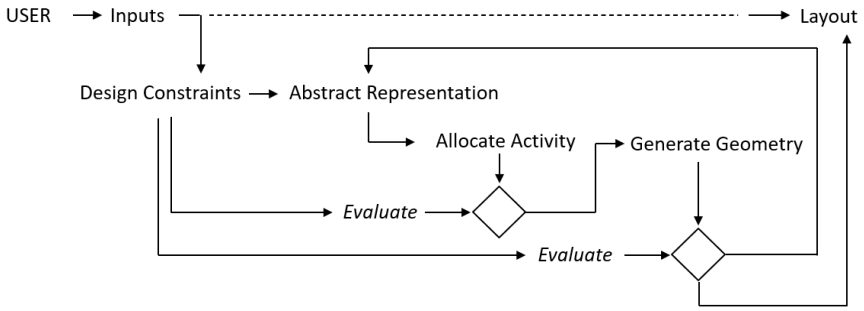


Figure 14 b. Program requirements of spaces

Figure 14. Arrays of space-activity relations and the processing

Spatial requirements of a layout are organized using reinforcement learning techniques. The location of spatial objects are altered until the necessary criteria are met (section 3.2.3). Each space of the layout is internally represented as a data object with attributes for the area, the curve that it occupies, name, unique identification number, values for the relationships between neighboring spaces, and cardinal directions. The location of spatial objects is based on access to orientation and adjacency requirements

provided by the user in the form of a spreadsheet (Table 9). Initially, the geometry of spatial objects is ignored and a point location for each object is scattered evenly across the region. Since each space has a unique identifier, the geometry is generated by associating a point location with the identifier. The geometry is generated by discretizing the bays into cells. Based on the sequence of spaces determined by the optimization process, cells are occupied by the space until the required area is achieved (Figure 15). An interpolation operation is required to ensure the numerical equivalence between requirements and the consecutive cell area.

Table 9. Program requirements of spaces

Spaces	Name	Area	Depth	Number	Color (rgb)	f0	f1	f2	f3	f4
0	f0	1000	100	2						1.0
1	f1	300	100	2				0.3	0.7	
2	f2	500	100	2						
3	f3	850	100	2						
4	f4	350	100	2						

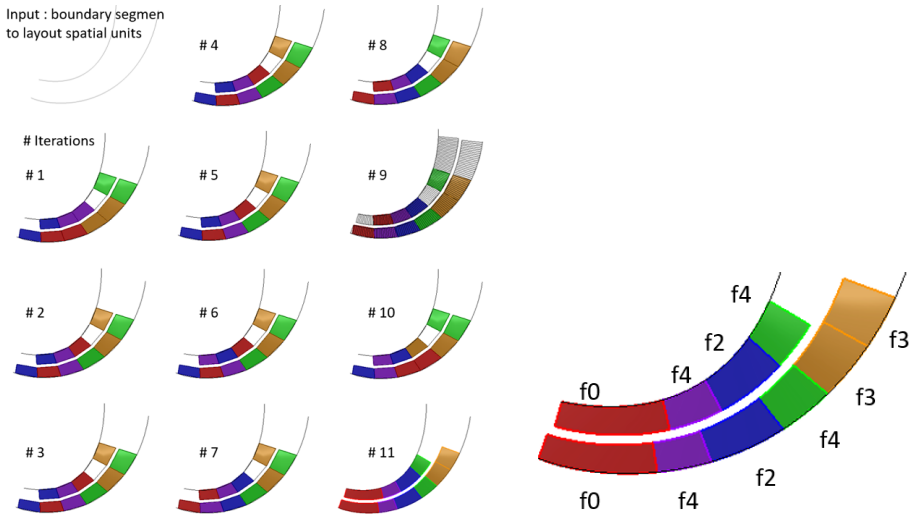


Figure 15. Iterations of the organization of spaces based on constraints above (Table 9) and the result below that satisfies the requirements.

3.4.3 Subdivisions of a curve

Partitions of the internal region of a boundary are used to separate the functional requirements of the floor plans. This is achieved by an equivalence relation between requirements and geometric operations on the closed geometric shape of the boundary. The relation is such that the shape is split in correspondence with cumulative areas of the two subsets of requirements. Recursively, the entire region can be split with a guarantee that it will meet the area requirements. These operations form a binary tree which can be manipulated to find relations between spaces by considering the configurations of the binary tree. The equivalence relation between the partition of numerical requirements and its bounding geometry makes it convenient to extract required relations from a set. This is mapped as a bijective function on geometric operations on the boundary (Figure 16).

Consider the set of area requirements $\boldsymbol{L}$={a, b, c}. An operation * on $\boldsymbol{L}$ is an unordered set formed by taking elements of two original subsets without repetition. Each set can be translated as the sum of elements i.e. area requirements.

$$A_i = \{x \in A: \forall x \in L, x \in A_i \text{ and } x \notin B_i\} \text{ and } B_i = L \backslash A_i$$

$$A_i \cap B_i = \emptyset \text{ and } A_i \cup B_i = L$$

The permutations can be dynamically generated to reveal variations, proximity relations, and the distance between two spaces. Once the geometric operations are defined, operations on set $\boldsymbol{L}$ can be manipulated to find relevant configurations. This is crucial to the development of optimization functions that will aid the refinement of the output.

The binary operation on ***L*** partitions it into subsets $\{A_i\}$, $\{B_i\}$. ***L*** can be partitioned in many ways, but d = min $\Sigma(A_i\text{-}B_i)$ can be used as a selection criterion to preserve proportions. Alternatively, d can be a range say p/q to q/p, n/2, or a random point in the set. From this, {M, N} can be found where $M=A_i$, $N=B_i$ when d_i is minimum. The binary partitions of ***L*** form a binary tree structure such that at each sub-division level, the sum will equal the total requirements. Set ***R*** is formed by recursive binary partition until M_i and N_i have only one element. The pendants are singleton sets whose order is equivalent to the number of required spaces and their sum will equal the area requirements ***L***. To generate architectural plans, the partition of the set ***L*** must be mapped to geometric operations. It must be a homeomorphic function and result in the formation of constituent polygons that correspond with the set of area requirements. This can be achieved by constructing a planar subdivision on boundary shape ***S***. It is an equivalence relation if the sum of each subset of partition ***L*** is used to subdivide ***S***. Set of requirements ***L***={a,b,c,...,n} gives a set ***R***={*L*, {M, N}, {M', N', M", N"}}.

Simultaneously, for a shape ***S***, the geometric subdivision should yield set of polygons ***Q*** = {S, {X, Y}, {X', Y', X", Y"}}, such that:

$|\boldsymbol{R}| = |\boldsymbol{Q}|$ and $\Sigma R = \Sigma$ Area(Q)

M o N = ***L*** and X o Y = ***S***

X= M and Y= N => X+Y= ***L***

The geometric subdivision is an equivalent of a binary partition of ***L*** because each time the set of the area requirements is partitioned, the sum of elements of the subsets can be used to split the geometry with a guarantee that area of each polygon will numerically match the required subsets. It takes the form of an equivalence relation based on the partition of set ***L***, area requirements.

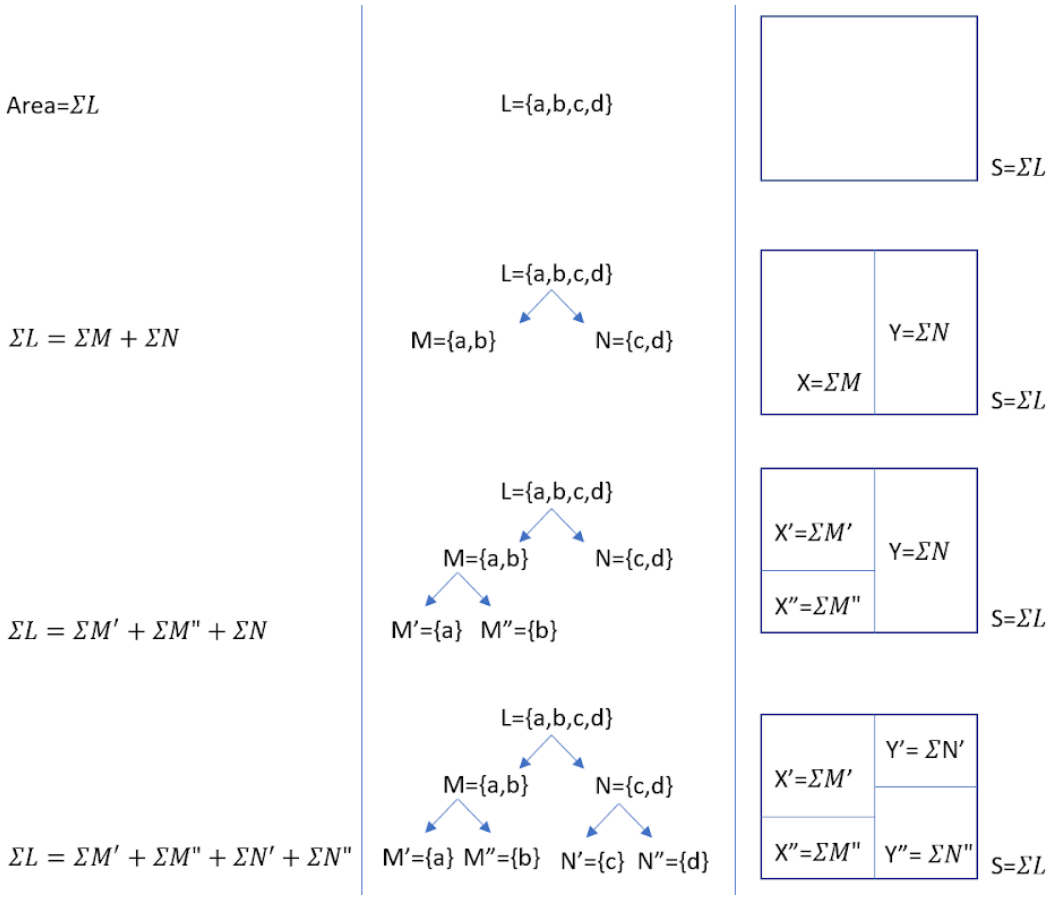


Figure 16. Binary Tree from requirements, uncontrolled partitions, and the resulting geometry

3.4.3.1 Controlling the partitions

The proportion of the subdivided polygons can be controlled by judiciously selecting an edge to conduct the geometric operation. For a rectangle, it is a choice between the long or short edge. For a general polygon or curve, the alignment is based on horizontal, vertical orientation, longest diagonal, or a spine generated by the medial axis.

An array is used to control the selection of the initial segment for each step of the recursion (Figure 17). This is a set of numbers that encode a feature such as a length, alignment, or area. It can be constructed independently of other aspects of the problem such as type of geometry, number of recursions, or requirements. The recursion stack references the designated index of the ruleset and selects the segment for processing.

The elements of the control vector are considered states. They are generated by a stochastic process. It is an ordered vector of numbers or characters that can be sorted lexicographically. This construct provides sufficient control over the generation of planar geometry. It is intended that with this array, and given the binary tree decomposition of area requirements, the geometry generated should be consistent and repeatable

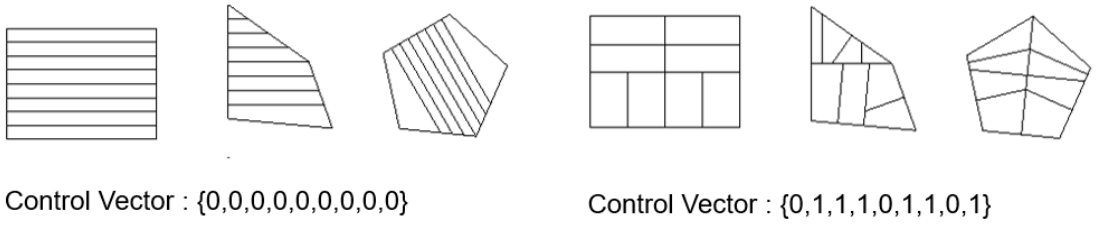


Figure 17. Partitions are controlled by vectors v: v∈{0, 1}, where 0:transverse, 1:longitudinal subdivision.

3.4.3.2 Adjacency Relations

Adjacency requirements are not addressed by the partitioning operations. Since the process of partitioning is represented as a graph, adjacency relations can be studied as sub-graphs. By altering the sub-graph, subdivisions are generated that ensure adjacency relations. These relations are addressed by altering the entire substructure from a node.

Two spatial objects are considered adjacent if they share an edge or vertex because they are linked physically. By scoring the number of adjacency relations, it is possible to continue altering nodes until the score cannot be improved. Over a number of iterations of alterning the position of spaces, the adjacency relations are met. The process relies on the input provided by the user. By adjusting the input values such as the strength of adjacency relation between two nodes, certain relations are prioritized over others. This ensures a maximum score based on the constraints provided.

A binary partition of the set of requirements ($\boldsymbol{L}$) can be mapped as a bijective function to generate a set of polygons that satisfy configuration and area requirements. The order of elements in the partition of the set $\boldsymbol{L}$ determines the final configuration. To find meaningful variations, the order is manipulated by shuffling the indices in a favorable direction. This causes a change in the binary tree of requirements. It controls the adjacency relations by reducing or increasing the separation between spaces. An artificial string is used (Figure 18) to conduct these operations. It eliminates the need for complex geometric manipulation, which is numerically intensive and non-intuitive. It is an algebraic construct that is easily visualized, parsed, appended, and provides a guarantee of uniqueness.

Since a binary partition produces two children, it is possible to construct a string which appends a character either ‘L’ or ‘R’ at the end of the parent string ‘O’. The data structure associates the string with elements of the area requirements and facilitates the manipulation of the binary tree using the swap operation. To find isomorphisms, child nodes of the tree are relocated. This alters the order of elements and changes the partition process. The change is reversible since the nodes can be easily interchanged if the result is not favorable. Using the strings, a parent node is taken as ‘OLL’, the children will be ‘OLLL’ and ‘OLLR’ and their children will be of the form ‘OLLL...m’ and ‘OLLR…n’. The entire subtree can be easily reconstructed or propagated by replacing ‘L’ with ‘R’ and ‘R’ with ‘L’ for any string length greater than the parent node string at an index equal to the length of the parent string. This process is illustrated in Figure 18.

Along with the vector to control geometric subdivision, the ability to consistently control the process of generation allows stochastic algorithms to dynamically fine-tune the parameters of generation. Although stochastic variables do not have the same value for different iterations, the underlying mechanisms described ensure that the user can alter inputs and control the generative process.

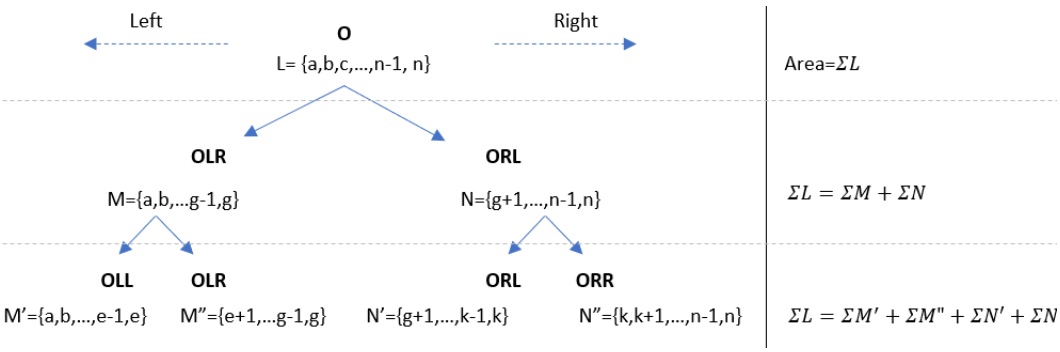
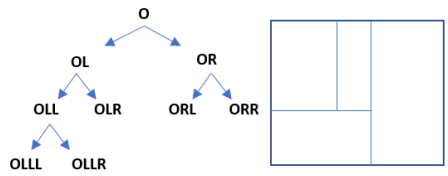
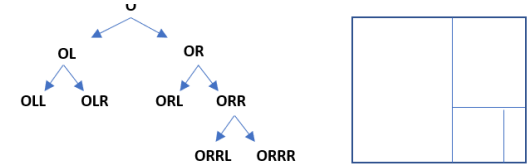


Figure 18 a: Enumerations based on isomorphisms

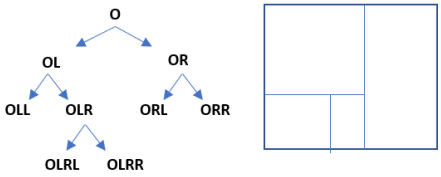
(Figure 18 continued)



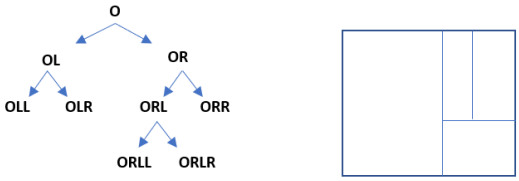
object: O --> ['OL', 'OR'] [1, 2, 3, 4, 5]
object: OL --> ['OLL', 'OLR'] [1, 3, 4]
object: OR --> ['ORL', 'ORR'] [2, 5]
object: OLL --> ['OLLL', 'OLLR'] [1, 3]
object: OLR --> [] [4]
object: ORL --> [] [2]
object: ORR --> [] [5]
object: OLLL --> [] [1]
object: OLLR --> [] [3]



object: O --> ['OL', 'OR'] [1, 2, 3, 4, 5]
object: OL --> ['OLL', 'OLR'] [2, 5]
object: OR --> ['ORL', 'ORR'] [1, 3, 4]
object: OLL --> [] [2]
object: OLR --> [] [5]
object: ORL --> [] [4]
object: ORR --> ['ORRL', 'ORRR'] [1, 3]
object: ORRL --> [] [1]
object: ORRR --> [] [3]



object: O --> ['OL', 'OR'] [1, 2, 3, 4, 5]
object: OL --> ['OLL', 'OLR'] [2, 5]
object: OR --> ['ORL', 'ORR'] [1, 3, 4]
object: OLL --> [] [2]
object: OLR --> [] [5]
object: ORL --> ['ORLL', 'ORLR'] [1, 3]
object: ORR --> [] [4]
object: OLRR --> [] [1]
object: OLRL --> [] [3]



object: O --> ['OL', 'OR'] [1, 2, 3, 4, 5]
object: OL --> ['OLL', 'OLR'] [2, 5]
object: OR --> ['ORL', 'ORR'] [1, 3, 4]
object: OLL --> [] [5]
object: OLR --> [] [2]
object: ORL --> ['ORLL', 'ORLR'] [1, 3]
object: ORR --> [] [4]
object: ORLL --> [] [1]
object: ORLR --> [] [3]

Figure 18 b. Generating the isomorphisms using the {'O', 'L', 'R'} notation.

Figure 18. Using a string to generate isomorphisms

3.4.3.3 Circulation in Binary Partition

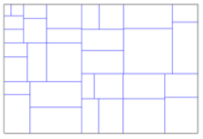
Circulation is generated from the partitions. For a given layout, the problem is considered NP-complete (Eastman1973; Shekhawat, 2019). From each partition line, a polygon is extracted with the user-input corridor width, aligned to the length of the partition line segment. The partitions are hierarchically sorted, and the region for circulation is

generated by taking half the corridor width in both the normal directions and performing a union operation to join them into a region bound by a single closed curve. To generate these regions for circulation, the process checks for intersection with all spatial objects. The spatial object has been developed with an attribute for connection. If this attribute is false, it is evaluated. Otherwise, the next object is queried. At the end of an iteration, all the objects are matched with a corridor. Due to hierarchical sorting, this matching is such that each corridor connects with the maximum number of possible objects and yields the minimum set of required corridors. If a corridor is connected to a spatial object, it is marked (attribute) as selected. Otherwise, it remains unselected.

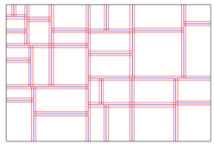
When a set of circulation polygons are joined (Boolean union), they must yield one polygon. However, this operation is not guaranteed. An iterative process is used to generate a connected circulation space. If there is more than one set of selected corridors, a minimum number of unselected corridors are used to join them. To achieve this, the disconnected circulation-polygons are sorted in descending order of maximum connections. The first corridor-polygon $\boldsymbol{A}$ is extracted and the remaining from a set $\boldsymbol{B}$. Elements of set $\boldsymbol{B}$ are extracted and joined to $\boldsymbol{A}$ with the minimum connections. Iteration over $\boldsymbol{B}$ is terminated when all spatial objects are connected. This is a greedy algorithm that ensures that all the disconnected corridors will converge into a single polygon $\boldsymbol{A}$ and remaining corridor-polygons in $\boldsymbol{B}$ can be discarded (Figure 19).

Initial input

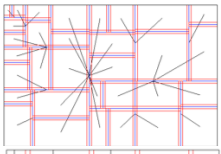
Binary Partition



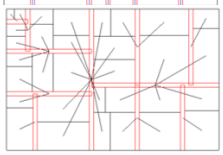
Generate Corridor polygon from each partition segment



Dictionary of circulation polygon with spaces connected to it in descending order



Greedily eliminate polygons that are not required



A solution of spaces with corridors

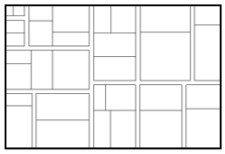


Figure 19 a. Steps to generate circulation

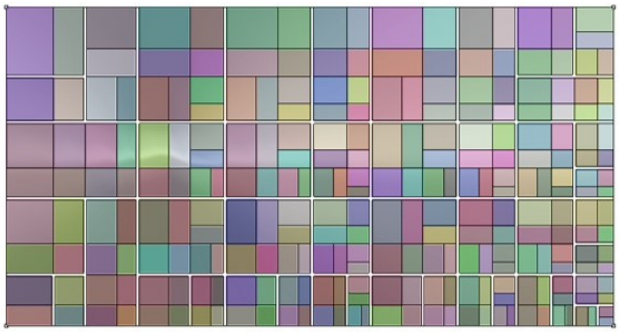


Figure 19 b. Scalable algorithms, large layout

Figure 19. Generating corridors

3.4.3.4 Organization: Connection to Optimization Module

Topological requirements are not addressed by the partitioning operations. Since the process of partitioning is developed as a tree, adjacency relations can be studied as sub-graphs. Altering the sub-graph provides an elegant approach to generate subdivisions that ensure adjacency relations because it is computationally inexpensive and allows intuitive control over the final layout. In the generative process, a binary partition of the set of requirements ($\boldsymbol{L}$) is mapped as a bijective function to generate a set of polygons that satisfy configuration and area requirements. The exact configuration depends on the order of elements in the set $\boldsymbol{L}$. The binary tree of requirements and the arrangement of spaces can be changed by shuffling the indices of $\boldsymbol{L}$. This mechanism is used to learn the optimal policy. It eliminates the need for complex geometric manipulation, which is numerically intensive and non-intuitive.

3.5 Specifics of Graph-based solvers (topological representation)

Spatial solutions with embedded design objectives or well-known principles of urban design and planning are generatated using mathematical models, where the user input enhances or suppresses the expression of design by manipulating the exposed design variables. These variables are in the form of numerical, boolean, or logical fields. They can be manipulated by the user at runtime. Although the output is in the form of a design and the aspect of problems and problem-solving techniques in architecture, urban design, and planning are highlighted, the primary premise of this research is the construction of

scalable models, where the same problem with a very large increase in the input is processed elegantly within a reasonable time. There are 4 parts of the space activity networks (Figure 20):

i. ***Parametric Grid:*** is generated with constraints for length, depth, cell length& depth.

ii. ***Network:*** It is initialized with 4 types of Vertices and 6 types of edges between them based on urban design objectives.

iii. ***Circulation:*** Geometry of streets that are generated from the edges of the network.

iv. ***Buildings / Space Allocation:*** Based on node type, nearest edges, and UI values for FAR, maximum height, etc. They are considered as containers of activity spaces to achieve area requirements such that type of building matches the facing edge.

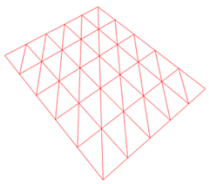
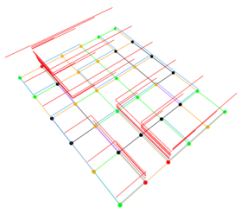
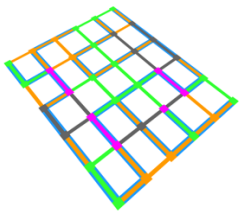
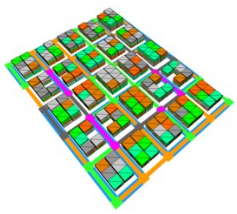
			
Parametric Grid	A network of nodes & edges	Hierarchy of circulation	Distribution of FSR and generation of built form

Figure 20. Showing the 4 stages of the space-activity network formulation. The buildings can be substituted by spaces of a layout and the circulation can be adjusted to represent floor plans instead of urban blocks.

The organization of nodes is based on the adjacency matrix described in section 4.2.2. The circulation is solved using the shortest path algorithms. Typically, Djikstra or Prim's algorithms are sufficient to find the shortest path or minimum spanning trees in a graph. The algorithms search for neighbors of a node and the sequence of a node with a minimum cost is accepted. Due to the presence of negative cycles, the Bellman-Ford algorithm is used (Table 10 a). It ensures the detection of the negative cycle and provides a path through the graph at a minimum cost.

Multiple objective functions are organized by constructing models with internally connected design variables. Instead of directly computing the cost using the edge weights, discounted rewards are computed based on the Bellman Equation. A table for the cost between any two nodes provides a reference to reach a goal from any given state. This form of learning is known as *Q-learning***,** commonly used in Reinforcement learning (section 3.1). It ensures that the agent will not try to maximize immediate rewards, greedily. Rather, cumulative rewards based on future states will be computed. This matrix, known as the *transition matrix,* constitutes the probabilities between states and taken together forms a *policy* that determines the best path from all nodes to reach the goal.

In this model, the urban space-activity networks are approximated by four interacting networks. Although the urban-problem formulation may utilize additional or fewer networks, these are typical in a mixed-use district-scale development. The topological representation of the network (Figure 21) is composed of:

a. Activity nodes that represent the activity type of spaces. They are distributed using an adjacency matrix. The four types of nodes assumed are:
    i. Pedestrian nodes for activity spaces such as residential, recreational, food.
    ii. Commercial nodes for offices, large commercial spaces where vehicle access is required.
    iii. Neutral nodes for services, public institutes where heavy traffic is not anticipated but possible.
    iv. Evacuation nodes for shelter from natural disasters.
b. Edges represent the circulatory structure of the built environment. By connecting various types of nodes (above), a hierarchy of circulation can be determined. The edge types are:
    i. Pedestrian – pedestrian, cycle, light scooters
    ii. Road – vehicular traffic
    iii. Neutral path – links green and road circulation
    iv. The intersection between green and road – connects green & road
    v. The path connecting all nodes – autonomous vehicle, bicycle track
    vi. Evacuation path from all nodes to EVAC node – quickest access from each point in the region

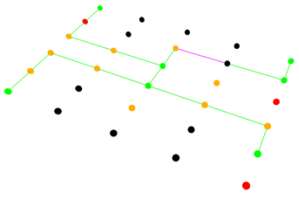
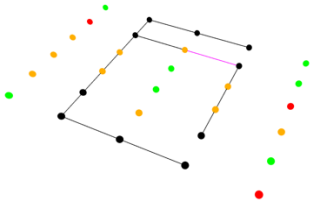
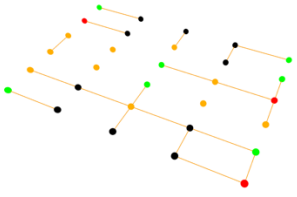
While the constraint-based routing and location of spaces is demonstrated in section 3.2, the main assumptions that generate the routing are:

i. Edges connecting the pedestrian circulation network and road circulation network must be mutually exclusive if possible. In case of an intersection, it must be displayed

ii. A neutral circulation network must avoid green and road edges and where possible provide a minimum cost.

iii. The minimum spanning tree must connect all the nodes with a minimum intersection with road or green edges.

iv. Evacuation must be the shortest path for each node and account for distance.

	Connections between GCN (green) nodes – mutually exclusive wrt RCN
	Connections between RCN (black) nodes – mutually exclusive wrt GCN
	Minimally connected NCN (orange) nodes with minimum intersections with GCN and RCN

(Figure 21 continued)

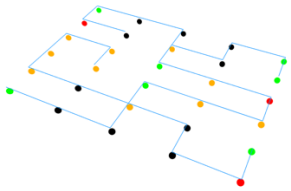
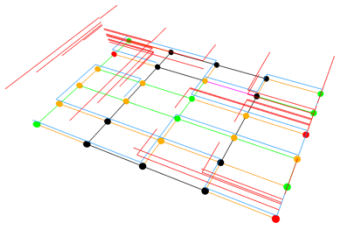
	The minimum spanning tree between all nodes – for autonomous vehicles or light scooters
	The evacuation from all nodes to evacuation nodes (red). And all the circulation routes taken together.

Figure 21. The illustrations demonstrate how the above-mentioned design objectives can be achieved and the scope for error in the form of intersections.

The edges are computed using the shortest path algorithms (Figure 21). The cost function of connectivity is exposed to the user and numerical rewards are computed as discussed in section 3.1 to generate the network and built form as illustrated in Table 11, Figure 22. From an input matrix, the cost of traversal between any two nodes is known. It changes the score and alters the objective functions as shown in Figure 10. A tabular form for the score is used to represent the values. By altering the cost function the user may change the interaction between networks. The shortest path is not strictly restricted to exclusive network connections. (Table 10 and 11, figures 21 and 22)

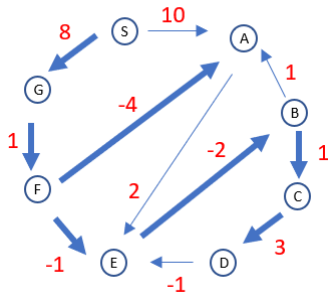
Bellman-Ford Algorithm:

Directed G(V, E)
Edge-lengths (l_e : e ∈ E) with no –ve cycle
Vertex s ∈ V
Output: ∀ u ∈ V reachable from s, dist(u)=dist(s,u)
For all u ∈ V:
    dist(u)= ∞
    prev(u) =nil
Dist(s)=0
Repeat |V|-1 times:
    ∀ e ∞ E:

        update (e)

Table 10 (a,b, c) . Shortest path algorithm & Centrality-dispersal algorithms

10. a computing the shortest path

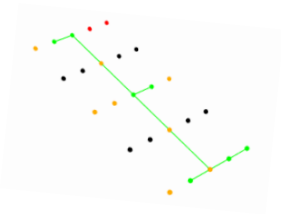


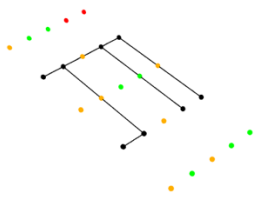
Node	Iterations							Parent
	1	2	3	4	5	6	7	
S=0	0	0	0	0	0	0	0	0
A=∞	10	10	5	5	5	5	5	F
B=∞	∞	∞	10	6	5	5	5	E
C=∞	∞	∞	∞	11	7	6	6	B
D=∞	∞	∞	∞	∞	14	10	9	C
E=∞	∞	12	8	7	7	7	7	F
F=∞	∞	9	9	9	9	9	9	G
G=∞	8	8	8	8	8	8	8	S

(Table 10 continued)

(b) Demonstration of cost matrix and the effect on circulation system or the interaction between nodes

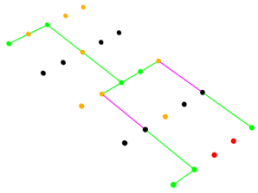
	GCN	NCN	RCN	Evac
GCN	0.10	0.30	0.75	0.95
NCN		0.45	0.85	0.95
RCN			0.95	0.95
Evac				0.95

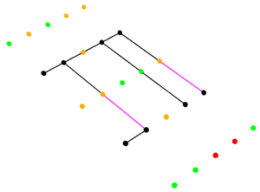




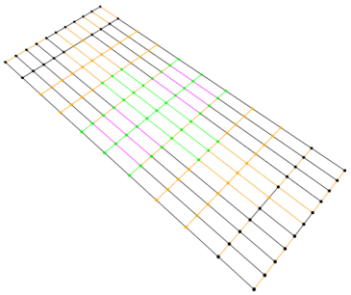
Altering the objective function to generate solutions with greater interaction.

	GCN	NCN	RCN	Evac
GCN	10.0	0.0	1.33	1.05
NCN		2.22	1.17	1.05
RCN			1.05	1.05
Evac				1.05

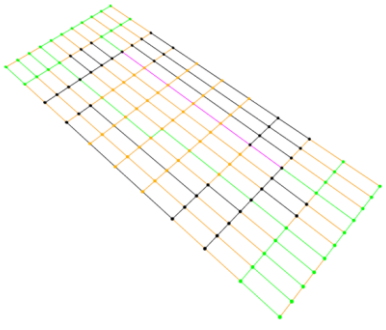




(c) Centrality-dispersal of nodes

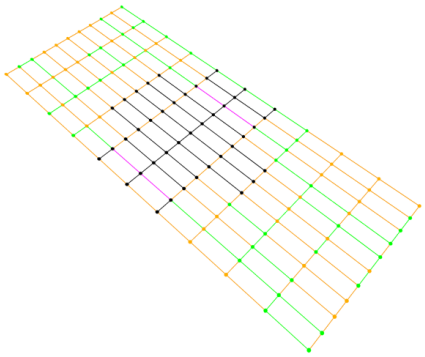
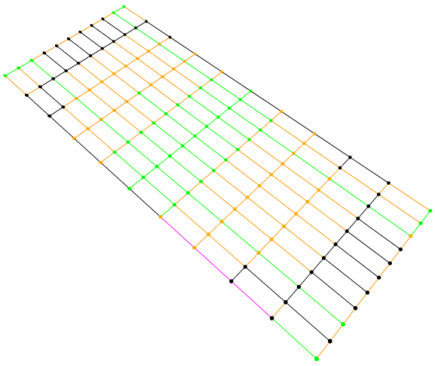
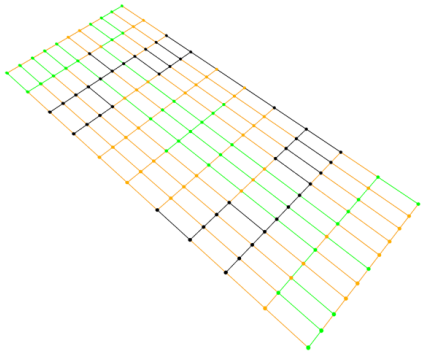
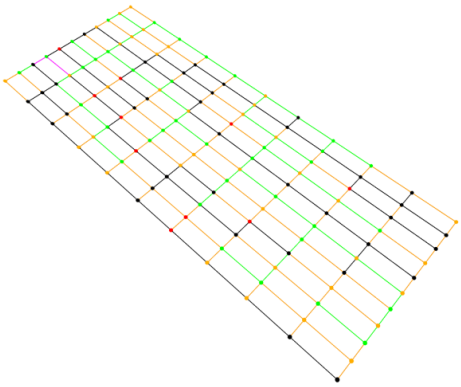


Centralize location of GCN Nodes



Centralize location of NCN Nodes

(Table 10.c continued)

	
Centralize location of RCN Nodes	RCN in restricted to end
	
RCN in restricted between center and end	Deliberate organization of the nodes and edges using cost functions and centrality-dispersal.

First, the location of nodes and edges of the graph (the connectivity between nodes) are organized, then the streets, and sites for further development of parcels are extracted and finally FAR is distributed to generate the buildings. The hierarchy of streets is based

on the type of edge. User input depth of streets is mapped to the edges and the street network is generated. The region enclosed by streets forms the sites or parcels. Floor space requirements of activity are distributed based on the nodes associated with each site. In the following illustrations, mixed-use development is used to demonstrate complex solutions. In such cases, the overall distribution of FSR is assumed across all the corresponding nodes. The node type accompanying the site determines the amount of activity applicable to the site.

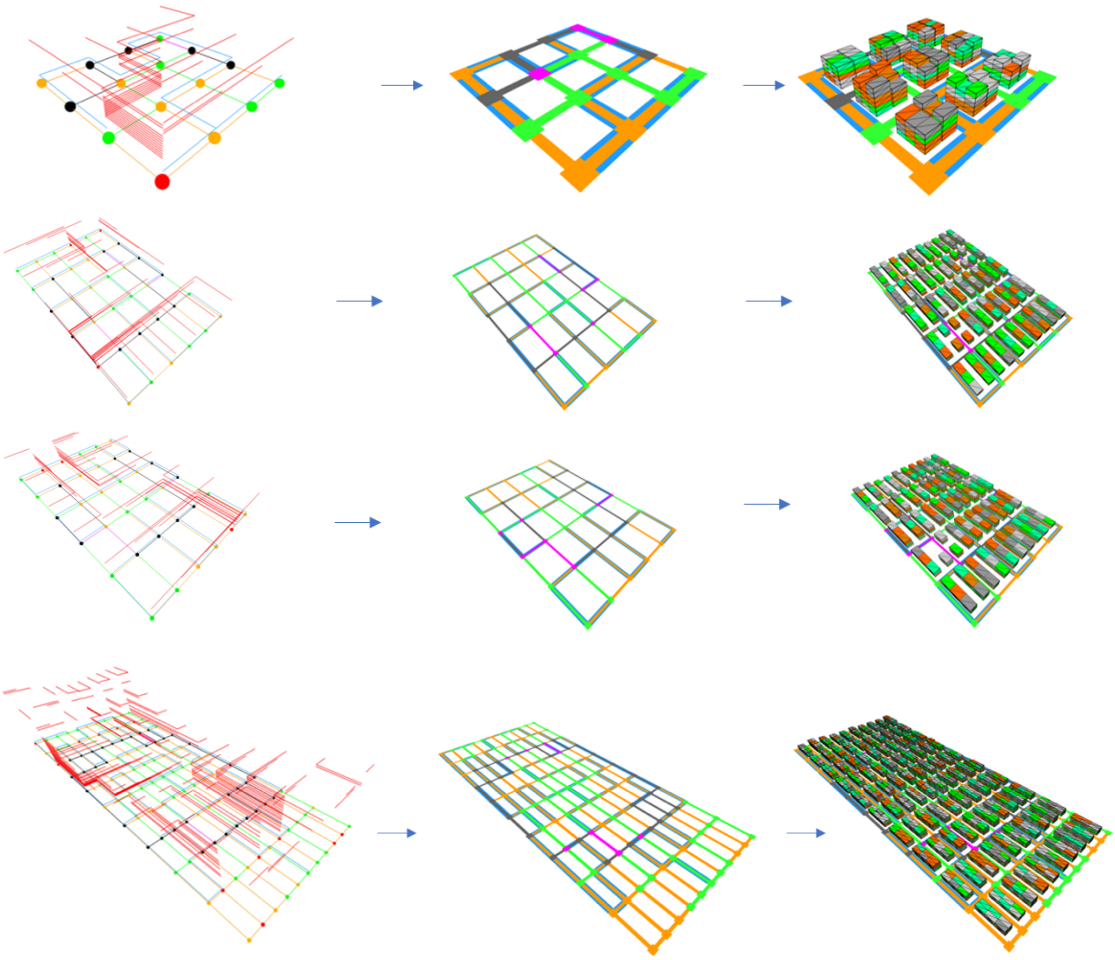
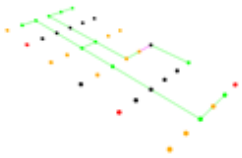
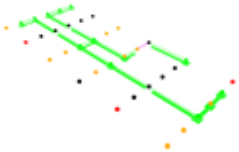
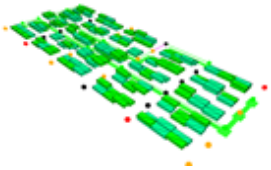
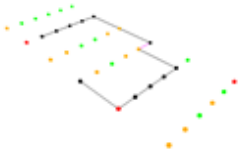
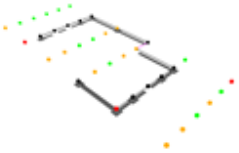
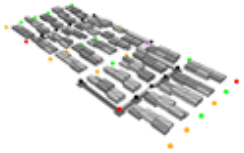
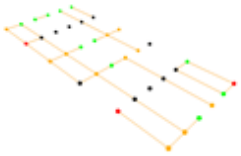
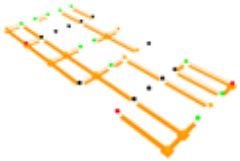
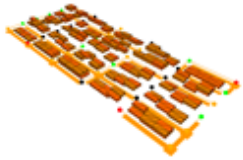
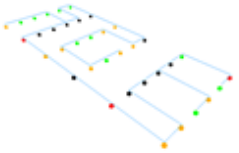
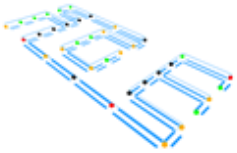
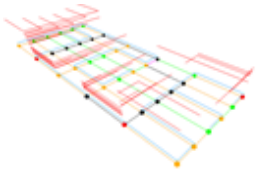
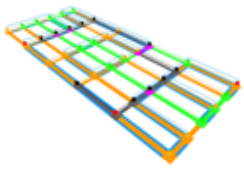
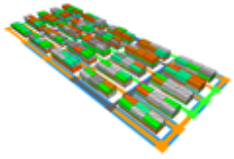
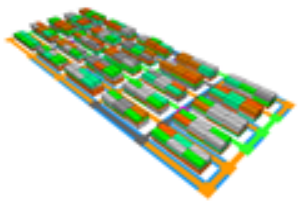


Figure 22. Sample of the generative process from networks to streets and buildings (activities)

Table 11. After placing the nodes and generating the circulation network, streets and FSR distribution are determined.

	Network Optimization	Street Hierarchy	FSR Distribution
Green			
Road			
Service			
Connector			
Evacuation			
Overall Solution			

3.6 Development Environment

This research in generative techniques led to the development of two major implementations in the form of distributable software. The space-activity networks for the city block layout (PLUGS) was initially developed as a web-based application. This was followed by a framework (IDF) of components and models which specifically focused on the SAP in design processes. It was developed in a way that allowed it to be associated with common tools used by designers and researchers in architecture and planning.

3.6.1 Integrated Design Framework (IDF)

A plugin for the Rhino-Grasshopper environment, namely, ***Integrated Design Framework***, or ***IDF*** is developed to support and validate this research. The software is experimental and consists of numerous components or computational models to solve SAP in design processes. The components can be connected in various permutations to generate elements of the built form. The components share data structures and methods. This facilitates the flow of information. The environment is chosen such that it can be used existing alongside analytical tools to provide a holistic solution. It is available at https://github.com/nirvik00/IDF

IDF has been developed as a direct result of supporting allied research who have a broader analytical interest in architectural and urban design (Yang et. al, 2020, Chang et. al. 2019, Rezzae et. al, 2020). IDF is supported by well-known software frameworks

(ladybug and honeybee) which allows designers with specialized knowledge to access them and develop project-specific workflows. It also makes it possible to disseminate knowledge and information.

The *IDF* framework of SAP components adopts a conjunctive method for developing workflows and conducting parallel explorations. Components for design entities can be used interchangeably. Once the computational pipeline is constructed by connecting the components of IDF, the user may modify the inputs and generate dynamically optimized results. The governing curves can be manipulated at runtime to reconfigure the entire scheme. They permit a real-time manipulation of bylaws such as setback, step back, maximum height, open space requirements, or selection of curves to be extruded. Apart from user-interaction, it is possible to introduce randomness into the system of generation. It is also possible to specify the amount of randomness. Since site feasibility is typically under constrained, a provision has been made to introduce a timer and automate the generation of design options. The implicit aim of this experiment (IDF) is to generalize the solutions and generate realistic formations of spaces found in practice.

3.6.2 Planning Urban Generative Systems (Plugs)

To demonstrate space activity networks, a web-based design generation, and visualization tool has been developed. It focuses on architecture, urban design, and planning. Standard web technologies such as MongoDB (no-SQL database), Express, and Node (web-server) have been used to develop the software. Three.js and datGui.js are used

for visualization and graphical user interface respectively. Python is used to transmit data from proprietary software such as ArcGIS and Rhino3d to the remote database. The web service was hosted remotely on *Heroku* using a remote database, namely, mLab. The web application is available at https://plugs-web.herokuapp.com/

The technology stack allows greater processing power. It enhances the reach of the software and permits collaborative participation from designers, casual visitors, and developers. This framework was used to assist in the development and assessment of urban planning requirements for a Japanese city (Yang P. et. al., 2020).

CHAPTER 4. COMPUTABLE MODELS OF SAP

This chapter describes the development of SAP-models to generate the spatial output of design processes, which is the primary objective of the thesis. SAT, discussed in the previous chapter namely, generation of geometric forms, optimization, or information processing are organized in specific ways and bundled into several models to facilitate intuitive usage. The mechanism of a typical SAP-model is described in detail along with the pattern of connections between them that lead to the development of an integrated framework (section 4.1). The proposed models are implemented in the form of software, namely, IDF and PLUGS. In the subsequent sections, the SAP-models are developed and illustrated by the solutions generated by the corresponding IDF components (sections 4.3-4.5). The concatenation of multiple spatial models to address SAP or workflow of connected components is described. Using object-oriented programming techniques, the IDF is developed where each model is mapped to a component of IDF (section 4.6). In the following chapter, using the features of SAP (section 1.4, p 18), the spatial output of IDF is demonstrated. The findings are used to determine the validity of the proposed solutions and the extent to which SAP problems can be solved using the proposed SAT.

4.1 Models of Space Allocation Problems

SAP is identified in design processes and broken down into computable tasks or functions. These tasks were mapped to elemental models (Figure 23 a) and auxiliary

functions that process input information and generate the solutions using embedded SAT. The SAT permits topological variance in problems and workflow of models can be applied to similar problems (Figure 23 b). These elemental models are concatenated to approximate SAP in large design processes across scales (Figure 23 c).

A design process may be interpreted as being composed of a sequence of generative steps where each step is approximated by an SAP and tasks of the SAP are mapped to computable models. In addition, auxiliary functions are provided for parsing inputs and connecting to optimization algorithms within the task. These additional functions assist in developing semi-automation features and allow real-time interaction. Using the models, a design process is interpreted as workflows of connected elemental models that approximate complex design processes by sequentially generating a set of spatial output that is processed by the next model (Figure 23).

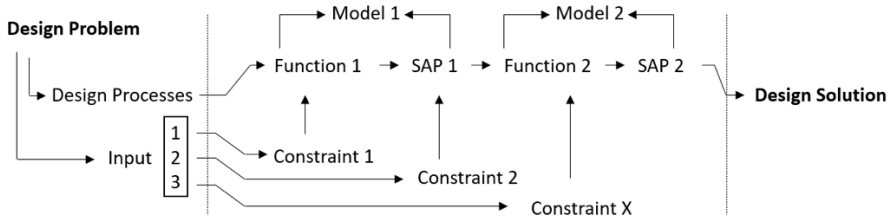


Figure 23.a. Extracting models from design processes to develop IDF components, where (a) SAP – Space Allocation Problem and (b) Function – Auxiliary function that collects and processes inputs to constraints.

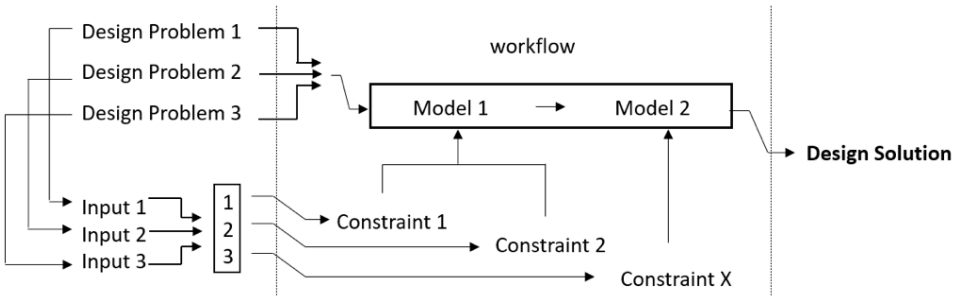


Figure 23 b. The same workflow of IDF Components (from Figure 2a) generates customized project-specific design problems.

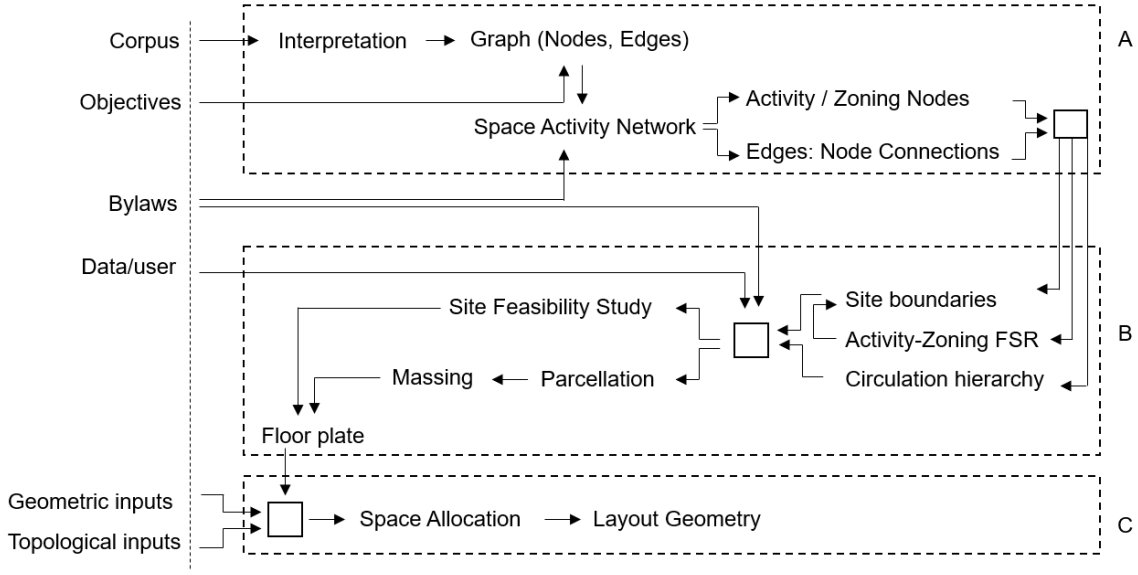


Figure 23 c. Schematic representation of the *linked spatial models* of design processes (Table-1): (A) Urban Layouts (B) Site Plan (C) Floor Plan

Figure 23. Overview of a computational framework to solve design problems

The internal structure of the proposed computational models consists of numerous processes (A-E) marked in Figure 24. A brief description is as follows:

i. Process-A *Interface*: These are functions that convert user-inputs into parameter values and constraints. These variables are embedded in the relation between the sequence of geometric operations that generate the architectural entity (process B). Constraints determine objective functions and evaluate the intermediate output. Users can interact with the model using the interface features.
ii. Process-B *Relations*: Logical and numerical relations with stochastic variables and parameters that control geometric operations and allocate activities. They receive inputs from process A and conduct operations process C.
iii. Process-C *Operations*: A pre-defined set of sequential geometric functions that operate on inputs of general geometric forms (process A) with parameter values from process B. The output is evaluated and errors or scope for improvements are determined.
iv. Process-D *Evaluation*: Based on the objective functions (process A), and the output generated by process C, an error is calculated or the score of the output is determined. This information is sent to the optimization algorithms process E. If the deviation of the layout is found to be within tolerance, the output is considered a solution to the problem. In this case, the loop is terminated.
v. Process-E *Optimization*: Topological optimization uses the error in a target function or score of the output to determine appropriate values for stochastic variables and parametric relations (process B) that control space allocation and geometric

operations (process C). Over several iterations, errors are minimized to generate appropriate solutions.

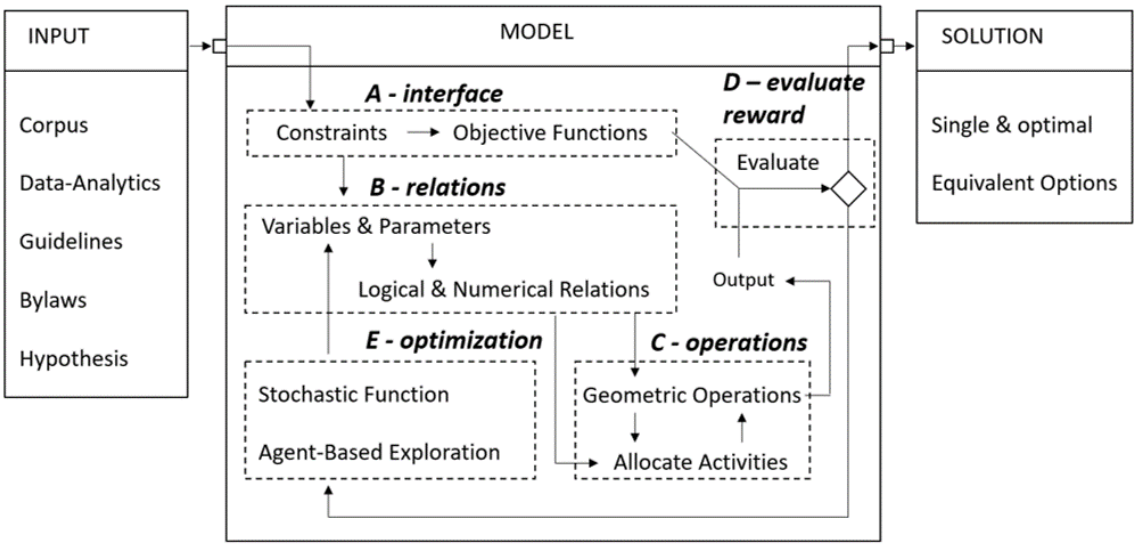


Figure 24. The generic internal structure of a model of the design processes. Processes (A-E) are described in section-4.1.2.

4.2 Constraints for optimization from Standardized Input Formats

Internally, the relations within the proposed models (figures 23, 24) are blocks of logical and numerical relations that guide the geometric operations of a specific design entity. Each elemental model is based on constraint-driven problems where the constraints are internally generated from inputs provided by the user. Solutions are generated from the recommendations and constraints developed by urban design methodologies, prescriptive rules, and guidelines that drive the computational models of the design processes.

From section 3.1.1, design processes commonly require solutions driven by optimization processes. It may lead to an exact solution with the highest score or numerous alternatives with an equivalent score within a range. The existence of multiple solutions is exploited in design exploration. Alternatively, the effect of constraints on a layout can be explored in order to determine the appropriate inputs that inform design decisions. It is being proposed that the spatial models are flexible such that they are used to find constraints and generate output from space-activity relations when it is known.

Based on constraints used in various prior research techniques (section 2.1.2), necessary and sufficient relations were identified and formats for organizing information were standardized. This standardization takes the form of an intuitive structure based on user interpretation. In this thesis, two formats of inputs are used (a) a matrix for constraints between elements (Table 12), and (b) matrix for properties of the element (Table 13). The former is parsed to determine the correlation between variables and the latter is used to assign attributes to an object (design element). The most common form of interaction between two elements is the grouping or separation between them. This is constrained by the adjacency matrix or a distance (proximity) matrix. The matrix is parsed by row-column indices. The desired attributes of elements or spaces are parsed row-wise. This captures the procedural geometric processes that generate the form.

For instance, in a hospital layout, a nurse station should be easily accessible to patient rooms. Alternatively, the lobby must be separated from radiation rooms. These inputs can be encoded as an adjacency matrix (Table 12). They are indicative of the attractive and repulsive forces between any two spaces. In addition, bylaws for fire-escape

may require staircases at intervals of linear distance along the corridor. Such design-objectives can be input by the designer in the form of an adjacency matrix, parsed as constraints, and used to determine the efficiency of the layout.

Table 12. General scheme for inputs for constraints between elements (Topological)

Matrix	a	b	c
a	V_{AA}	V_{AB}	V_{AC}
b	V_{BA}	V_{BB}	V_{BC}
c	V_{CA}	V_{CB}	V_{CC}

The geometric attributes of spaces such as area or dimensions are also accepted in a tabular form but parsed differently (Table 13). These attributes may be constant such as area required or preferred orientation of space. This information is used by the variables and parameters of the relations between geometric operations in the model (Figure 24). Arrays of values for a parameter are acceptable inputs. When the input interface detects multiple values, conditional functions are invoked and the optimization process operates on the arrays. Using string tokenization, the interaction between these parameters is unraveled and assigned to appropriate fields. The constraints may be set up in a way to reference nested constraints presented to the algorithm in a different file. This allows the organization of information and aids decision-making or evaluation of the output.

Table 13. General scheme for inputs to parameters of elements (Procedural)

Element	Parameter 1	Parameter 2 (conditional)	Parameter 3
a	P_{A1}	P_{A20}, P_{A21}	P_{A3}
b	P_{B1}	P_{B20}, P_{B21}	P_{B3}
c	P_{C1}	P_{C20}, P_{C21}	P_{C3}

A generative process operates on the attributes and allows designers to experiment. Each design process has specific types of inputs (Table 14). They can be manipulated to generate the desired organization of the spaces. Inputs that generate the constraints for optimization may be informed by the corpus, data-analytics, local conventions, and hypothetical arguments. Inputs can be adjusted to alter the constraints and study behavior of the model. They can be used to study alternatives generated as a result of the propagation of constraints.

Table 14. Problems and their standardized inputs

Type of Process	Inputs (converted to constraints)
Space Allocation	Program of requirements, adjacency matrix (prior art)
Site Feasibility	Program of requirements, vertical adjacency matrix
Parcellation and Massing	Bylaws/user-preferences
Space-activity Networks	Adjacency & centrality matrix for nodes, cost matrix for edges or circulation

4.3 Space Planning (floor plans or building layouts)

The input interface embedded in space planning models generate constraints from the inputs (Table 15) topological requirements and geometric requirements. The generation of geometry and allocation of spaces are treated as separate processes in this research (Figure 25 a). This allows greater efficiency in arriving at an exact solution. The optimization process takes the form of a search for locating the spaces and assigning an activity to them. It is ensured that each iteration improves the solution (section 3.1.3). Whereas, the shapes are generated by procedural geometry algorithms. By separating the geometric form, it is possible to treat it with sophisticated methods and replicate commonly found shapes in architectural plans.

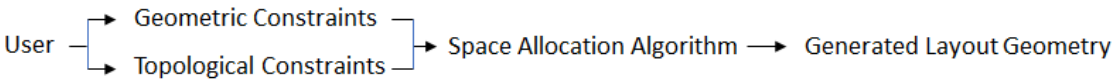


Figure 25 a. Space allocation using geometric, topologic input to generate constraints

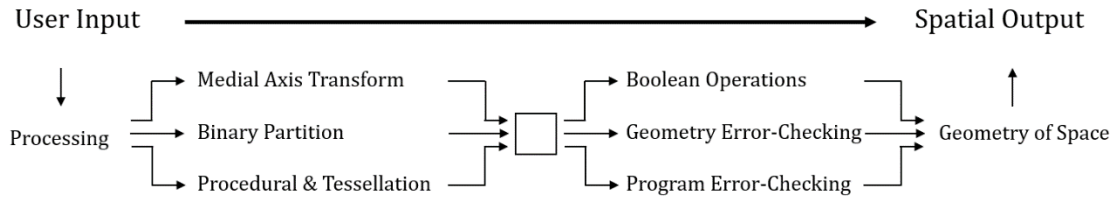


Figure 25 b. Geometry generation algorithms

Figure 25. Geometry Generation

The proposed solutions take into account a flexible process of design is where designers can continuously change the requirements and update the geometry (Liggett, 2000). Effectively, constraints are explored during the process of design. A mechanism for continuously updating the layout has been developed (Figure 26) where the proposed solver uses methods to propagate constraints and understand conflicts. It determines whether the sub-structures or existing partial configurations of the layout have maximized their objectives or minimized the cost function (Figure 26). This ensures that optimal sub-structures will be retained. The location of spaces is updated sequentially. Over several iterations, an appropriate configuration of the layout is determined.

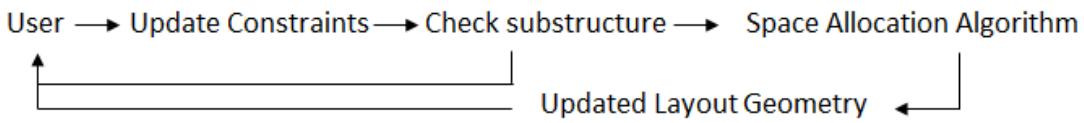


Figure 26. Space Allocation in a changing environment

Table 15. Sample of user inputs that are converted into constraints for space planning

Spaces	Geometric constraints		Adjacency Constraints					Input
Names	Area	Number	a	b	c	d	e	Color
a	150	1	0	0	12	0	0	
b	500	1	0	0	0	0	0	
c	400	1	0	0	0	0	0	

An overview of the model that emulates the space planning design process (Figure 27):

1. The model is initialized with user inputs of adjacency and program.
2. Inputs are converted into constraints between elements and attributes of spaces.

3. The organization of activities and the geometry of spaces for each activity are separated and parameters are passed to the respective functions. A topological model of the layout is maintained with provisions for activity and geometry. It is updated as the location of spaces and the corresponding form is determined.
4. Reinforcement learning algorithms based on agent-environment interaction is used to calculate substructures using stochastic processes. It updates the layout without disturbing an optimal substructure. The process terminates when an alternative position of space cannot be determined such that it improves the score.
5. Geometric operations are conducted to determine the actual geometry of the spaces. This is based on the desired characteristics or attributes of the spaces.
6. Circulation is generated to connect the spaces.
7. The geometry of spaces, their activities, and the circulation forms the layout.

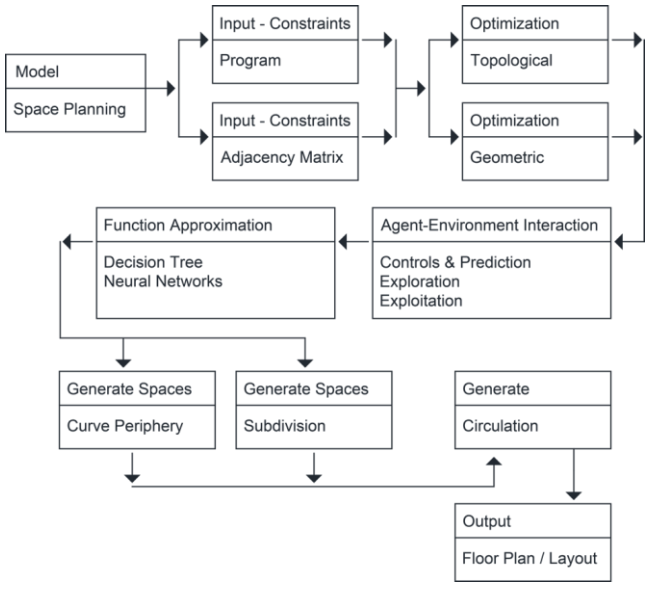
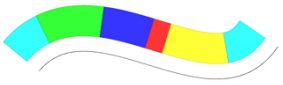
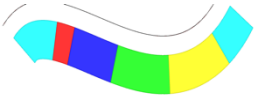
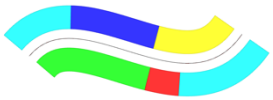
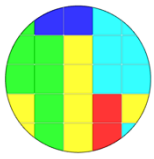


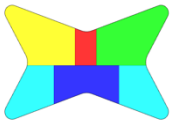
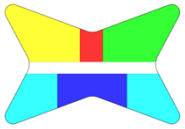
Figure 27. Scheme of space planning models

Implementation in IDF: The following spatial components are proposed as models for space planning. Their identification is based on a unique organization of spaces and the circulation that accompanies such configurations (Table 16):

Table 16. Components for Space Planning.

SL	Symbol	Solution	Details
a		Side 1  Side 2  Center 	*Purpose: Spaces along a curve.* Constraints from Table-2 Sub Category: Space Allocation Attract: {a, c}  Repel: {e, e} 
b			*Purpose: Spaces on either side of an open curve.* Constraints Table-2 Sub Category: Space Allocation
c			*Purpose: Spaces along the periphery of a closed curve.* Constraints: Table-2 Sub Category: Space Allocation
d			*Purpose: Grid-based space-allocation* Constraints: ***Table-3*** Sub Category: Grid Repel: {a, c}  Attract: {e, e} 
e			*Purpose: Generate & partition a rectangle.* Constraints: **T**able-2 Sub Category: Space Allocation

(Table 16 continued)

f			*Purpose: Space allocation n a closed curve.* Constraints: Table-2 Sub Category: Space Allocation
g			*Purpose: Circulation for spaces generated in (f)* Constraints: (f) Sub Category: Space Allocation

4.4 Site Plan: Parcellation-Massing, Site Feasibility Studies

The site plan permits two distinct approaches (Figure 28). These are parcellation-massing and site feasibility studies. Both processes output building mass and their floor plates. The following sections illustrate the development of models for these processes.

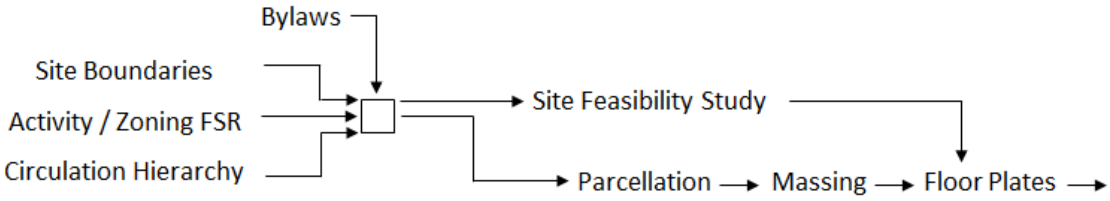


Figure 28. A dichotomy in site plan development

4.4.1 Parcellation-Massing

It is proposed that elemental models of building-mass and site parcellation can be concatenated and used to generate a wide variety of solutions for parcellation (site

subdivision) and subsequent massing (volumes of building mass). Activity allocation can be applied to the mass by extracting floor plates. This allows these models to connect with the space planning model (section 4.2.2.1). The parcellation-massing model (Figure 29) processes include the following steps:

1. The parcellation-massing is initialized with inputs.
2. The inputs are converted into constraints for internal geometric relations.
3. The site is recursively partitioned into parcels.
4. A building-mass typology is chosen, and volumes are generated to meet the area.
5. The building form is generated based on the specifications provided.
6. The floor plates are generated for each mass.
7. Each floor plate can be input for space planning models.

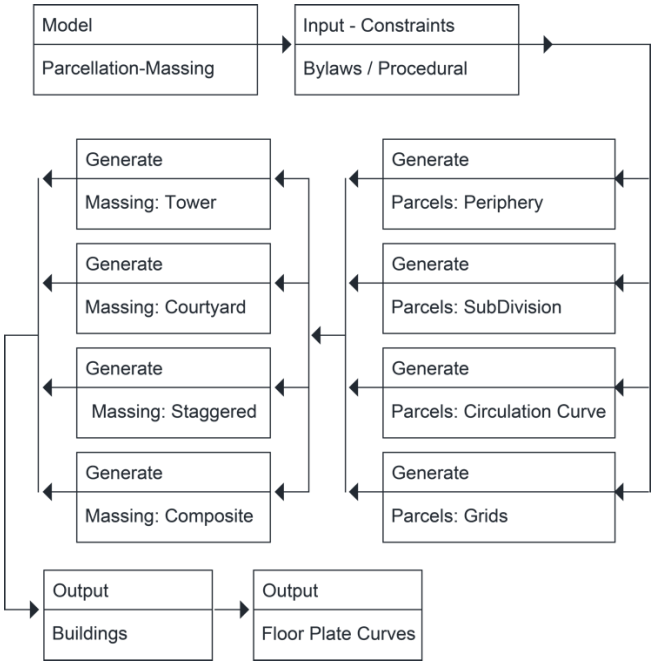
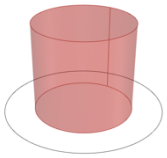
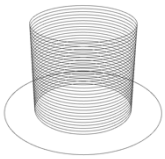
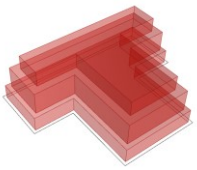
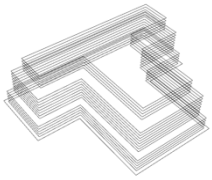


Figure 29. Scheme of parcellation-massing models

Implementation in IDF: Commonly used formal typologies of building-mass have been treated as a procedural-model. Their attributes can be directly accessed at runtime. These components generate the building footprint from a site boundary or parcel and extrude them according to required height parameters. The area requirements are governed by numerical relations between the boundary and vertical growth. Building surface and floor plate curves for each level are output. Massing typologies are shown in Table 17. The methods for space allocation can be used to generate parcels for building extrusion. Apart from these modules, typical partitions used by designers are shown in Table 18.

Table 17. *IDF*: Massing Studies to support parcellation.

SL	Symbol	Massing with Floor Curves	Floor Plate curves: calculate the area
a		Setback and extrusion (tower typology), 	
b		Setback, step back extrusion (stepped-tower typology) 	

(Table 17 continued)

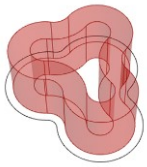
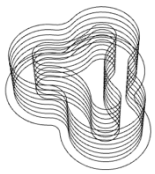
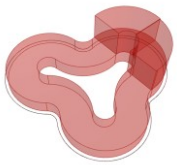
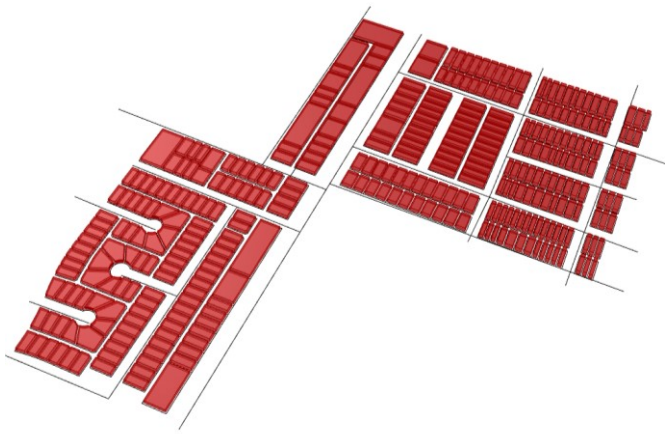
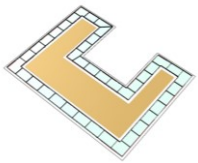
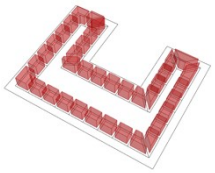
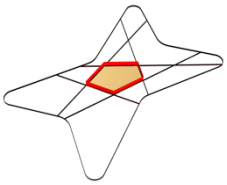
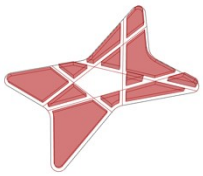
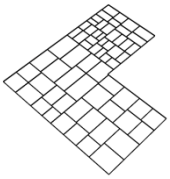
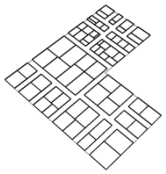
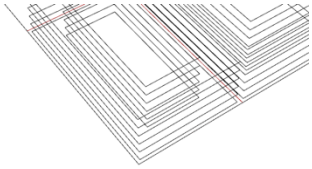
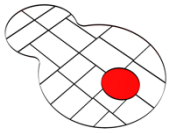
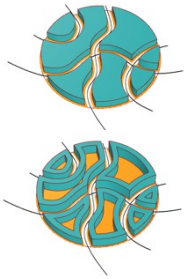
c		Setback, bay depth, and extrusion (courtyard typology) 	
d		Podium and selected stepped-tower extrusion (courtyard typology) 	
e		Mass setback-step back extrusion for a network of parcels and streets 	

Table 18. *IDF*: Provision for general partition exploration to support site feasibility studies.

	Symbol	Site Parcellation	Symbol	Massing from parcels
a		Generate circulation and parcels along the periphery of a given curve 		
d		Determination of a partition-scheme from an internal polygon 		
b	 	Generate circulation and parcels in the interior region of the input curves  Include circulation: 	 floors	 
c		Generate internal parcels which exclude specified regions 		

(Table 18 continued)

e		Parcellation based on circulation curves 		

4.4.2 Site Feasibility study

While parcellation and massing are exploratory in nature, a site feasibility study is conducted to accommodate program requirements. Computationally, programs become non-trivial when various types of buildings are introduced along with proximity relations, dimensions, and activities for the collection of floor plates generated by the buildings. Bylaws for the height, setbacks, stepbacks, open space requirements increase the complexity of this problem. For such a problem, stochastic processes are devised to support the design process or site feasibility studies. In this study, buildings are placed on a site and certain activities are allocated to the floor plate based on their elevation or proximity. This is a form of space allocation where the buildings are separated by distances rather than adjacencies. The model provides a method to generate appropriate configurations of buildings and activities within a range of inputs provided by the user. Summary of the site feasibility study model (Figure 30):

1. The model is initiated with distance matrix and program requirements.

2. The inputs are converted into constraints between elements and properties of the elements. The footprint of each building type is constrained by proximity relations between them (Table 19). The activity type, their building type preferences, area, and height requirements form the second type of inputs (Table 20) required to guide the site feasibility model.
3. The building footprints are found by heuristics.
4. A building typology can be chosen and applied to meet the area requirements.
5. Floor plates are generated from the floor-height and FSR requirements.
6. Activity is allocated to the floor plates.
7. Each floor plate is sent as input to the space planning model.

Table 19. Sample of inputs: building details, proximity/distance matrix (Part 1).

Name & Identification		Geometric Constraints				Distance Constraints			
Name	Key	Number	Footprint Area	FSR	Ratio	a	b	c	Color
name_A	a	2	100	0.5	0.75	12	12	5	
name_B	b	1	100	0.2	1.25	5			
name_C	c	3	100	0.2	0.75	5		12	
name_D	d	2	100	0.35	1.25	5			

Table 20. Sample of Inputs: activity details for the site feasibility study (Part 2).

Identifier		Location Constraints		Building Identifier				
Name	Key	Percentage	Height	Parent	Parent	Parent	Parent	Color
name_A	p	20	50	a	b	c	d	
name_B	q	70	-	a	c	-	-	
name_C	r	10	200	a	b	-	-	

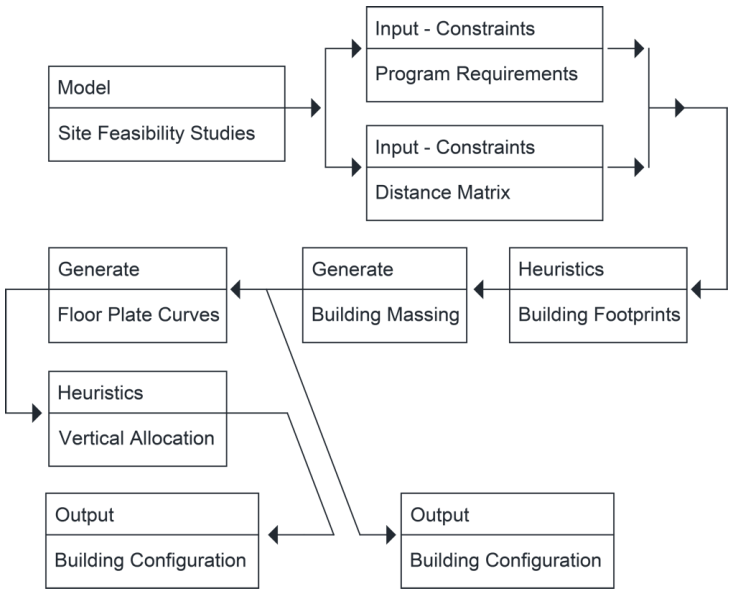


Figure 30. Scheme of site feasibility study models

Implementation in IDF: The site feasibility study has been treated as a nested problem. First, the building footprints are located based on a proximity matrix and geometric attributes (Table 21). To accommodate bylaws and area requirements, the building mass is generated along with floor plates. Activities (zoning) are allocated to the floor plates collectively using the constraints for building types, elevation, adjacency (Table 22) and the location of buildings may be constrained by excluded areas of the site or height restrictions. Also, open space requirements and specific height constraints are addressed by the optimization process. It is an indirect consequence of the generative process. (Figure 31)

Table 21. Building Details, proximity/distance matrix

Name & Identification		Geometric Constraints				Distance Constraints			
Name	Key	Number	Footprint Area	FSR	Ratio	a	b	c	Color
name A	a	2	100	0.5	0.75	12	12	5	
name_B	b	1	100	0.2	1.25	5			
name_C	c	3	100	0.2	0.75	5		12	
name_D	d	2	100	0.35	1.25	5			
The distance matrix performs is similar to the adjacency matrix, unspecified values are assumed as 0									

Table 22. Activity Details for allocating activities to floor plates

Identifier		Location Constraints		Building Identifier				
Name	Key	Percentage	Height	Parent	Parent	Parent	Parent	Color
name_A	p	20	50	a	b	c	d	
name_B	q	70		a	c			
name_C	r	10	200	a	b			
The building type (parents) is used to identify the building to which activity is allocated.								

Figure 31 a: Plan: Location of space extents to place building footprints based on Table 21 above, excluded regions & site extents.

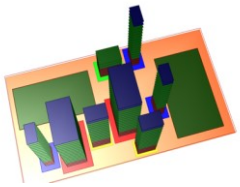
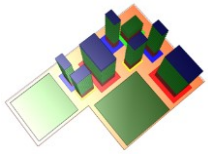
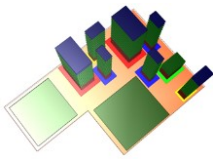


Figure 31 b: Perspective View- Generation of floor plates from the building footprint based on FSR requirements (Table 22).





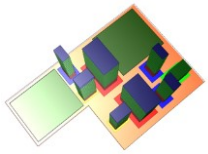


Figure 31. Sample of alternatives for over-constrained problems. Illustration of various configurations of excluded regions and automated solutions.

4.5 Interactive Space Activity Networks in City-Blocks

Lynch (1988) proposes that a city can be represented as a graph with nodes of activities and circulation as edges (Figure 32) where the *functional theory* would allow the graph to be solved using mathematical programming. This is similar to the graph-theoretic formulations (section-2.3) of the floor plan.

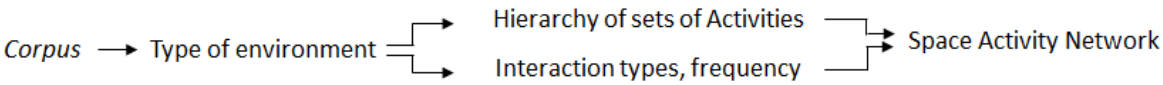


Figure 32. Generating space-activity networks from the corpus and data

In this thesis, it is proposed that the space-activity network allocation model allocates activities across multiple sites. A topological model of the urban layout is used. It consists of several networks that interact with each other. The interaction governs the relative location of nodes and edges concerning elements in the same or a different network.

The nodes of each network are distributed by distance from the center of the graph. They are organized by an adjacency matrix. The edges or connection between any two nodes are generated by a cost matrix (Tables 12 and 13). The edges are determined using modifications of the shortest path and minimum spanning trees. This is a generic computational model that is presented to designers for modification.

To create layouts from the network of space-activity relations, the network is transformed into parcels, streets, and buildings (Figure 33) by mapping the topological

structure to a closed curve(s). The percentage of total FSR determines the activity allocated to nodes of each type. Similarly, the edges of the space-activity networks represent the hierarchy of connections utilized in the generation of the circulation system. Sites for parcellation are generated from the region enclosed by the streets. These are inputs for subsequent design processes. It is a layered problem based on graph theory and computational geometry.

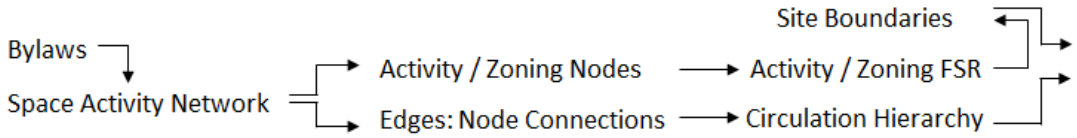


Figure 33. Space activity networks are resolved into site boundaries and circulation system.

The interactive networks of space-activity relations in city-blocks as a model is implemented as the distribution of parcels along with the area of activities and the generation of a circulation system between various types of parcels. This is a generic model that can be modified according to an interpretation of existing data, the corpus, or the preferences of the user. The space-activity networks model includes the following steps (Figure 34):

1. The model is a topological representation of urban layouts. It contains several sub-graphs. They are initialized by converting adjacency and cost matrix inputs into constraints.
2. The location of each network is determined by centrality and the cost matrix is used to connect two nodes to form edges between networks.

3. Within each sub-graph, the nodes and edges are organized by an adjacency matrix and cost functions.
4. The edges lead to hierarchies of street networks.
5. Sites for parcel-massing and subsequent space planning are generated from the nodes and streets using a convex hull algorithm.

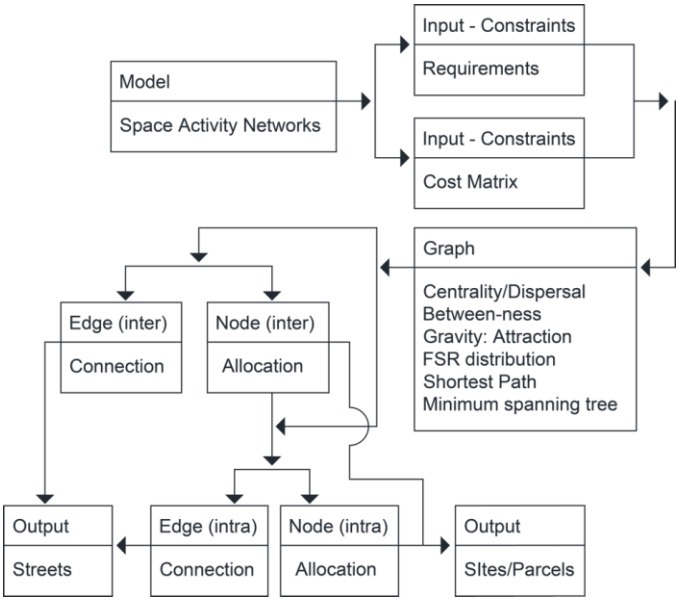


Figure 34. Scheme of space activity network models.

4.5.1 PLUGS: (Problem 3) Interactive Space Activity Networks in City-Blocks

Space-activity networks are treated as numerous interacting subgraphs. The model is a graph G (V, E) which is composed of sub-graphs $H=\{G_1(V_1, E_1), G_2(V_2, E_2), G_3(V_3, E_3),...G_n(V_n, E_n)\}$. The position of nodes is determined by distance from the center. The edges are connected by a matrix of cost requirements.

The design is generated from the model by four layers of processing (Figure 35) starting with a topological grid from which a network is extracted. This network is optimized for a variety of edge connections, frequency, and distribution of nodes. It is converted into a circulation layer consisting of roads and streets. Buildings are generated from the network based on calculations on the distribution of the types of activities. Several ways to constrain or stochastically explore design generation have been demonstrated in this research. The parameterization of objectives through a graphical user interface enables a general framework that is suitable for various kinds of design-context. Optimization of these objectives includes changing the location of activities, ensuring appropriate area to host activities in buildings, and reconfiguring the connectivity of the circulation network.

i. *Grid*: The grid decomposes a region into a set of cells (Figure 35 a). Each cell is bound by nodes and edges. This generates the graph or network based on which other layers may be generated. It is controlled by constraints for length, depth, cell length & depth.

ii. *Network Layer:* The network consists of nodes and edges, that represent activities and circulation, respectively (Figure 35 b). The mode of circulation in this context may fluctuate from pedestrian to vehicular. The elements of circulation in the network are responsible for providing access between the green routes and roads. A minimum spanning tree is used to connect all the nodes, assuming that it may enable cycling, autonomous vehicles, or similar light vehicular modes of travel. The network optimization includes:

a. Routing algorithms to find mutually exclusive vehicular traffic and pedestrian zones. Pedestrian routing is the set of nodes commonly associated with simple circulation such as connecting residences to a nearby shop for daily needs. Typically, it serves people within a locality. Vehicular routes connect large offices or commercial complexes commonly used by people outside the region of study or locality.
b. A set of edges that minimally connect all the nodes of the region.
c. A set of maximally connected streets that permit bicycle tracks, routes for autonomous vehicles, or light traffic.
d. Evacuation routes from each node to the nearest node for evacuation.

iii. *Circulation Layer*: From the optimized network, a circulation layer is procedurally generated (Figure 35 c). This generates the geometry of roads and pedestrian paths that connect their respective nodes and the path between them to permit easy access between the mutually exclusive routes. Sometimes, it is not possible to avoid intersections between pedestrian paths and vehicular routes, but undesired intersections are minimized and marked for review.

iv. *Buildings & Activity Layer*: The nodes of the network are associated with the activity types required (Figure 35 d). Their frequency, heights, and distribution can be controlled by the GUI. These attributes are reflected in the buildings generated. The region enclosed by a set of edges is used to either generate the built form or processed using site plan components (section 4.5.2). The nodes associated with the region determine the floor area distribution of the activity. Following the

assumption that multiple zones are preferred, the buildings follow a principle of vertically stacking areas pertinent to each activity type.

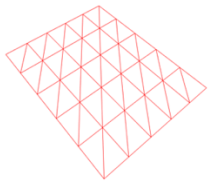
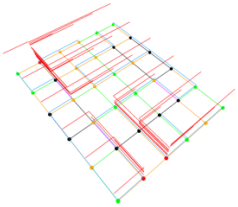
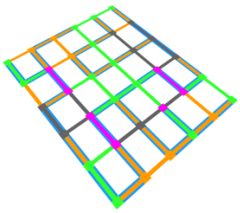
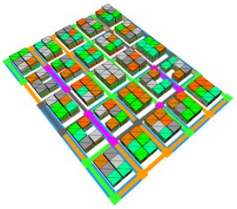
			
39 a. Parametric Grid	39 b. A network of nodes & edges	39 c. Hierarchy of circulation	39 d. Distribution of FSR and generation of built form

Figure 35. The 4 stages of the space-activity network formulation.

4.5.2 IDF: Attractor-based Generators

The previous section demonstrates the SAT to generate solutions for a given grid. In this section, the models for generating the grid and allocating FSR distribution from points and curves are demonstrated (Figure 36, Table 23).

There are two primary techniques to develop a street grid. They can be generated by a greedy algorithm that grows from a point. Or, given an existing grid, edges are selected based on constraints. Once a street-grid is formed, spatial attributes of the region are determined by allocating the floor-space-area distribution.

Various types of activities may need to be grouped or separated along a circulation route. This can be interpreted as an interaction between multiple networks. Minimization of travel time and maximization of FSR distribution are primary objectives in the design of networks (Peng, 2016). The travel time in a network is minimized by a densely connected street grid. This reduces the area for spatial activity on the plane. Such conflicting objectives require an optimization process.

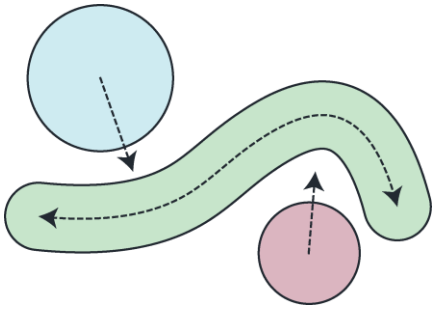


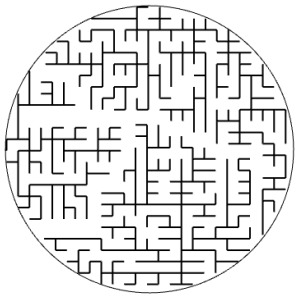
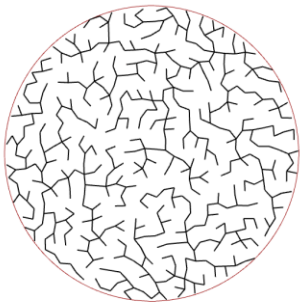
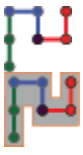
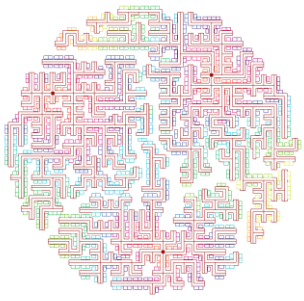
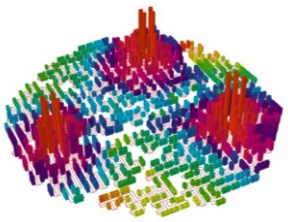
Figure 36. Generate grids from point-based or curves as the source of origin.

Components for complex grids and typical orthogonal and radial grids with *embedded space allocation algorithms* have been provided. They interpolate between a point and a curve or between two curves to generate the parcels and sites for development and distribute the FAR. A general technique for street networks develops the circulation for these spaces by greedily generating circulation segments with constraints for intersections, proximity relations, and a pre-defined range of length. Apart from the optimization of proximity relations, distribution of area by extruding parcels have been shown. The floor space distribution is driven by a gradient of distance from user-input points and curves. Multiple points or curves can be used to generate the FSR distribution

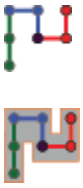
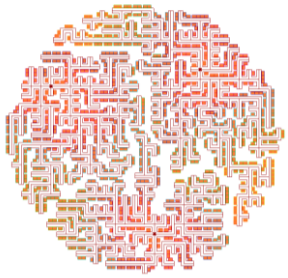
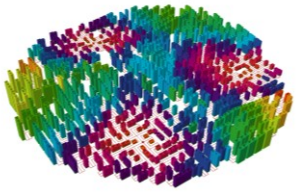
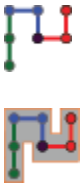
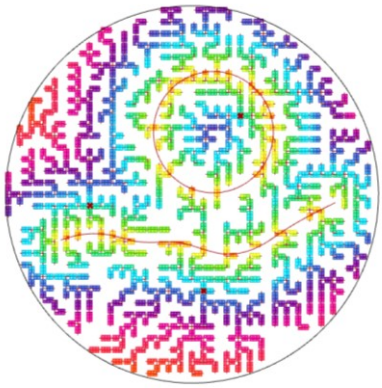
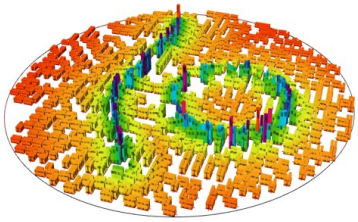
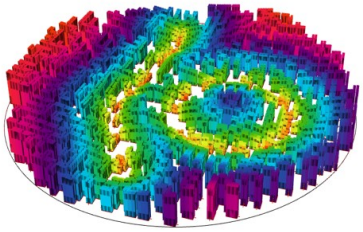
as a gradient of distance from the control point. The UI provides numerical controls to fix the minimum and maximum heights. Based on this, the gradient is normalized. This ensures that the output always generates a height within the domain.

IDF provides modules to generate networks of spaces. Since the techniques are scale-less, they can be used as nested functions to successively generate elements of the built environment from a network of spaces to massing of buildings and their floor plans.

Table 23. Point / curve-based street generation and Distribution of floor-space-ratio.

Symbol	Layout	Symbol	Large scale FSR allocation
	Point-based Rectilinear		Controlled stochastic functions
			
Point-based FSR distribution:			
			UI option 1: Maximize ht from points: 

(Table 23 continued)

<table>
<tr><td>  </td><td>  </td><td>  </td><td>UI option 2: Minimize ht from points:
  </td></tr>
<tr><td colspan="4">Curve-based FSR distribution for multiple curves, both open and closed can be used to guide the FSR distribution.</td></tr>
<tr><td>  </td><td colspan="3">  </td></tr>
<tr><td>  </td><td>  </td><td>  </td></tr>
<tr><td></td><td>UI option 1: FSR increases when parcels are closer to the curve boundary</td><td>UI option 2: FSR decreases when parcels are closer to the curve boundary</td></tr>
</table>

4.6 Elemental Models to Framework of Connected Models

During the development of the SAP-models, the flow of information was mapped across the input/output of models to connect them. Each model is a self-contained processing unit that can be used by a designer to assemble a workflow of models to develop spatial solutions for an elaborate design problem (Figure 37). For instance, by connecting a set of models, designers can rapidly prototype layouts for a building or conduct a site feasibility study for urban design and development (Figure 23 c). Permutations of workflows can be applied to replicate complex design processes in the practice of architecture and planning. The framework is open-ended such that additional modules may be introduced by identifying new patterns and problems in design processes. The framework is designed to support a potential flow of top-down (scale-wise) sequential SAP in design processes. The elements of the framework are:

i. *Variables* of the hard-coded relations in the models are exposed to the user. The user controls how these inputs influence the objective function and subsequently the design generation. They may be exposed to the user by a GUI.
ii. *Design variables* are embedded in the IDF components and represent attributes of the elements of the built environment such as dimensions and types of roads and streets or types of activities located in various buildings. They may be exposed to the user by a GUI or through standard input formats using excel files,
iii. *Design principles* that are considered axiomatic and the extent of their influence on other elements upon propagation and their parameterization is achieved by the SAT

and embedded within the components. The variables to control them are exposed as *design variables*.

iv. *Prescriptive rules* and numerical requirements which influence attributes of elements including geometry generation, frequency, dispersal, or adjacencies can be encapsulated in the proposed constraints (Table 14). They are parsed from the input interface and constraints are generated to guide the design principles.

v. *Model-free optimization* methods are used to determine the correlation, location, and quantity of elements such as circulation and activities (chapter 3). This is the optimization technique linked to a large number of modules where required.

The computational mechanisms provide a permutation-based approach- it is agnostic to actual design processes or man-made rules. Theoretically, it may be argued (Green, et. al, 2019) that the artifice of grid is applied along with zoning and planning considerations, thereafter, the grid is converted into buildable-closed curves (blocks and parcels) and hierarchy circulation. The development of an integrated framework is based upon Batty's hypothesis (2016) that cities are "*constellations of interactions, communications, relations, flows, and networks*" where the specific location of spaces and is a result of such interactions and incites a chain reaction. The application in an actual design process can flow in any direction and various permutations of components are possible, by supplyig inputs to the appropriate fields.

Figure 37 is a *class diagram* where the methods are shown in terms of initialization and flow of information across the modules. This means that suppose only one input is provided, a large piece of land, then starting by laying down a grid, the models support

design processes downstream to floor plans. In practice, there are many inputs (constraints), provided intermittently. The framework is designed such that the components are used in various permutations across the classes illustrated in Figure 37.

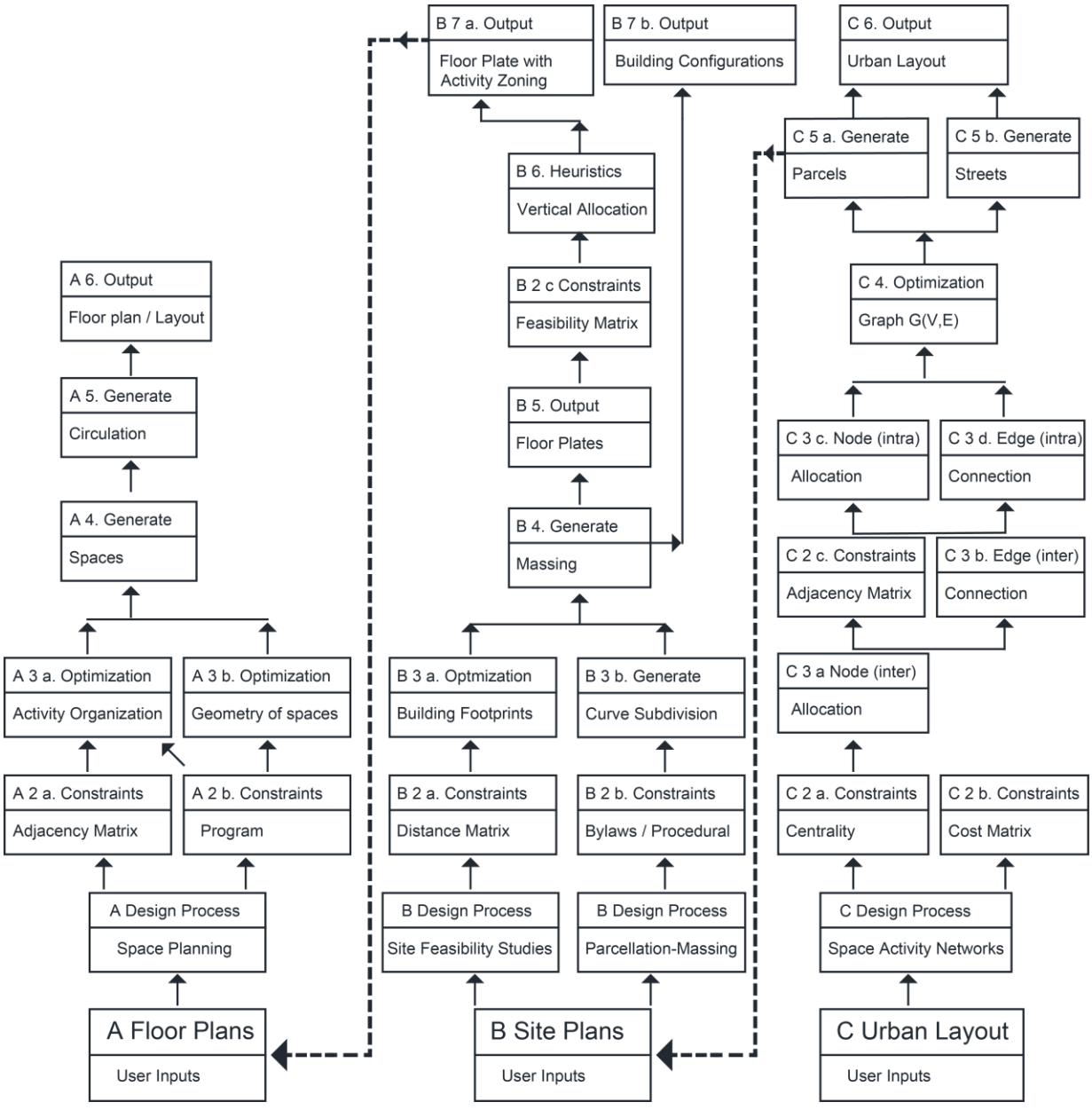


Figure 37. Scheme of an integrated framework of connected models (sample).

CHAPTER 5. INTEGRATED DESIGN FRAMEWORK

The previous chapter describes a set of general models that generate spatial solutions of SAP in various design processes and the accompanying software (IDF) to implement them. The solutions generated from the models are illustrated in this chapter. The key features of space allocation (table-2, p 18) are used to determine the efficacy of the proposed SAT, spatial-models, integrated framework, and the application of workflows to compute spatial solutions. Case studies and test cases are developed to determine the capabilities of the system and ascertain the extent to which objectives could be fulfilled. The case studies and test cases demonstrate various types of exploration that can be systematically generated using IDF workflows. The illustrations presented in this chapter explain the hypotheses that led to the formulation of IDF. The various types of systematic design exploration are examined in the next chapter.

Four case studies, based on problems found in the practice of architecture and urban design, are presented in this chapter to illustrate the application of IDF workflows to architecture and urban design projects. The case studies demonstrate the application of IDF in solving the four problems that were examined as objectives of this research.

1. Case Study A for space planning – demonstrates the application of IDF in the development of space planning in a healthcare facility with a complex geometric form and nested constraints between spaces. The geometric input is the floor plate boundary composed of polygons and curve segments. It is zoned into a peripheral band of spaces and four internal zones with a core for vertical circulation.

2. Case Study B for a site feasibility study – demonstrates the site-level exploration for residential towers where building footprints are provided as ranges of length and with, their separation distances and FAR distribution.
3. Case Study C for site parcellation-massing / site feasibility – demonstrates the exploration of a hierarchy of streets and building formations with block and shaped typologies that conform to existing urban morphology. The organization of spaces vertically in a tower and between towers is constrained.
4. Case Study D for urban networks – two explorations of urban space-activity networks are demonstrated. Using the existing footprints, activities are re-allocated based on various FAR distribution provided by the user. An alternative organization of parcels, open spaces, and mixed-zoning is demonstrated.

The test cases illustrate the step-wise development of a systematic solution to complex geometry and optimization techniques that are applied to architecture and urban design. The test-cases differ from case studies because they demonstrate greater flexibility of form and user interaction to test the algorithms.

1. Test Case A to demonstrate peripheral optimization in space planning – the optimization process is broken into two parts. First, spaces along a peripheral band are organized, then spaces are generated inside the curve and the overall configuration is optimized. This test case demonstrates the use of alternative components and shows the step-wise changes in the spatial configuration based on user interaction.

2. Test Case B for parcellation & massing exploration – demonstrates the application of curves and projecting the architectural typologies into various forms taken by the parcels as a result of their geometric interaction.
3. Test Case C for parcellation & massing exploration and decision-making – is extended by incorporating decision making components that inject alternative building typologies to generate a self-correcting mechanism.

The features of SAP-models (Table 2) that lead to semi-automation and provide an environment for the explicit design process are illustrated by the case studies and test cases. The standardization of design objectives led to the consolidatation of a wide variety of design problems that could be elegantly solved using a few IDF modules. Generalization of geometric form (section 5.2) is illustrated by case study A, which is a space planning problem. Optimization of various SAP (section 5.3) is illustrated by Test Cases A, B and Case Study B. The input interface (section 5.4) is a parser which parses user input and generates the objective functions or constraints to the optimization algorithms. The user interaction (section 5.5) between the user and the optimization algorithm is illustated by Case Study C and Test Case A. The application (section 5.6) of the algorithms is ported to a web-application to solve problems across the three scales are illustrated by Case Study D. The features above lead to a semi-automation framework (section 5.7) illustrated by Test Case C. Finally, the models are applied to illustrate the application of IDF in the generation-evaluation processes (Section 5.8).

5.1 Constraints as Design Objectives

(Can a wide variety of objectives be integrated?)

The space allocation problem is traced across three fundamental design processes (section 1.2, p 4). The proposed SAP-models in sections 4.3-4.5 are addressed by specific constraints (Table 14, p 140), where the constraints are a processed (mathematical) form of the objective function of the optimization process.

The constraints vary for each problem in two ways (a) the values and number of constraints change while the form of constraints provided to the model remains the same and (b) the *types* of constraints are different for each SAP based on the objective of the design problem being addressed. The generalization of models depends on the ability to process constraints from inputs provided by the user. For each problem, specific constraints are expected. The constraints are connected by internal relations built into the models where the resolution of constraints determine the numerical attributes of the geometry of spaces leading to favorable design solutions. The user inputs have been devised such that they can be intuitively manipulated by the designers. Essentially, the pipeline (Figure 38) allows the user to prescribe objectives to the model through inputs.

The inputs are based on prior information, guidelines, and recommendations. Or, hypothetical inputs can be used to generate solutions for further analysis. It was found that three types of constraints (objectives) were required:

i. *Topological constraints* are used to evaluate the layout. These are correlations between spaces that are unique to the design problem. They require iterations to determine an optimal solution. For instance, the adjacency and proximity matrices used to generate layouts and site feasibility studies.

ii. *Geometric constraints* determine the attribute of the spaces. They are user-defined dimensions or areas of the spaces that can be achieved by discretization of the curve. They do not require iterations but the upper and lower limits affect the accuracy of the solution.

iii. *Procedural constraints* that contain information about groups of geometric forms. These are exposed variables that can be altered by the user at runtime. In the algorithm, the variables are bound by mathematical relations. They do not need iterations, rather, they are solved by following a sequence of operations. For instance, the parcellation-massing models constrained by bylaws.

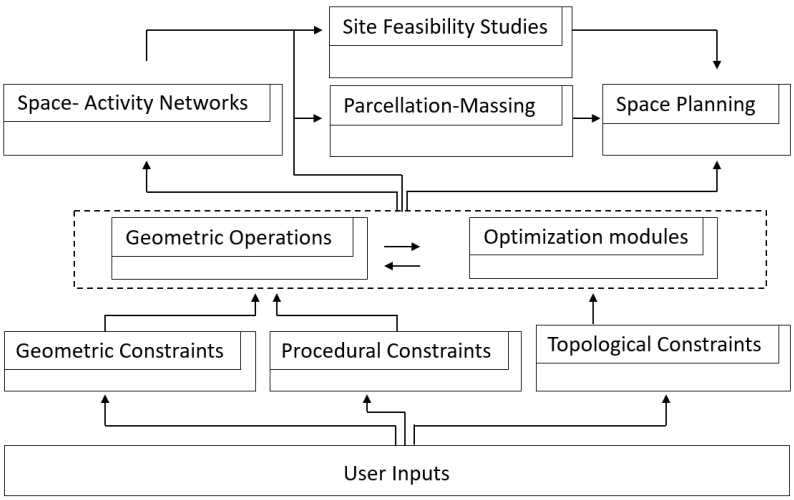
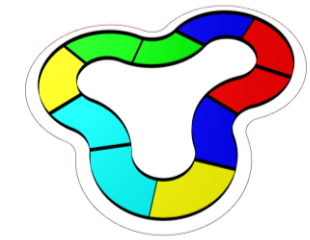


Figure 38. Organization of constraints, geometric operation and problems

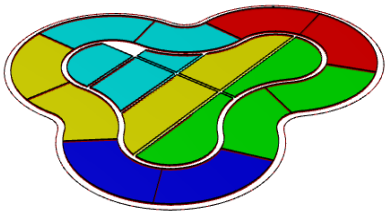
5.2 Generalization of Geometric forms

(Are the solutions generally applicable to all geometric forms?)

Typical architectural layouts are composed of spaces around the periphery and internal subdivisions, generalized to general curves and polygons. Such patterns can be demonstrated by IDF (Figure 39). The proposed models use geometric operations or SAT (section 3.4) in various ways to generate the appropriate output as shown in Case Study A.



39 a. Peripheral band of spaces



39 b. Subsequent Internal partition of spaces

Figure 39. General solutions for peripheral & internal organization of spaces – Test Case B

5.2.1 Case study A

This case study illustrates the generalization of geometric operations in practice. The original plan was created using a conventional design process and using the same inputs as designers, a large number of alternatives were developed using an IDF workflow. The study demonstrates the ability to generalize the geometric operations that lead to project-specific customization. The design problem was originally solved for a star-shaped composite shape (Figure 40). After replicating the solution, the workflow was applied to

diverse forms. These studies use the same constraints as the designers (Table 24-28) but require minimum human intervention to generate output while allowing dynamic updates.

The problem exhibits complexity in geometric form and constraints. But using IDF, a massive number of alternatives were generated (Table 29) and the workflow was extended to other primitives (Table 30, 31) that allowed the workflow to be tested for convex, non-convex, curve, orthogonal shapes – without alterations in the workflow. The salient features of the problem (Figure 40):

a. The unusual geometric shape including composite shapes, convex, non-convex and smooth corners
b. Varying organization of spaces at periphery, quadrants, and center.
c. An extensive list of requirements of the real plan (tables 24-28).
d. IDF workflow to test diverse forms because *generalization implies customization* (Tables 29, 30, 31)
e. Independent and alternative types of design exploration are demonstrated.

Figure 40. GIVEN geometry and requirements without spatial color-coding; zones (departments) are given a gross color.

Table 24. User-input for the peripheral set of spaces in Case Study A

List of requirements: Neuro Patient Room, Patient Room, Ada Patient Room, Conference, Training Room, Neuro Monitor Room, Reading, Interview Room, Ptot Therapy, EVS, Consult, Seating Area, Family Lounge, Staff Lounge, Workstation, Meds, Clean Supplies, Staircase, Nour, EVS, Linen, Shared Office, Public Tlt, IS, Dir Office, Trash Recycle, Pts, Elec, Treatment Room, Respite Room, Food Gallery

Names	Id	Area	Number	pA	pB	pC	pD	pE	pF	pG	pH	pI	pJ	pK	pL
neuro_patient_room	pA	1	14	100										0	
patient_room	pB	1	43		100					200				200	
ada_patient_room	pC	1	6			200								200	
conference_training_room	pD	1	4				400	100	100						
neuro_monitor_room	pE	1	1				100	0	100						
reading_interview_room	pF	1	1				100	100	0						
ptot_therapy	pG	1	1		200					200					
Evs	pH	1	2												
Consult	pI	1	3				100					400			
seating_area	pJ	1	1				100								
family_lounge	pK	1	1		200	200									200
staff_lounge	pL	1	1											200	

Table 25. User input for spaces in core-zone in Case Study A

Names	Id	Area	Number	core-A	core-B	core-C
Lifts	core-A	1.5	1		100	100
Zone_Left	core-B	1	1	100		-200
Zone_Right	core-C	1	1	100	-200	

Table 26. User input for spaces in zone-1 in Case Study A

Names	Id	Area	Number	z1.a	z1.b	z1.c	z1.d	z1.e	z1.f	z1.g
Workstation	z1.a	1	2	-200	100	50				
Meds	z1.b	1	1			50				
clean_supplies	z1.c	1	1			50	100	50	50	50
Staircase	z1.d	2	1				0	100	100	100
Nour	z1.e	0.75	1				100		100	
Evs	z1.f	0.75	1					100		100
Linen	z1.g	0.75	1						100	

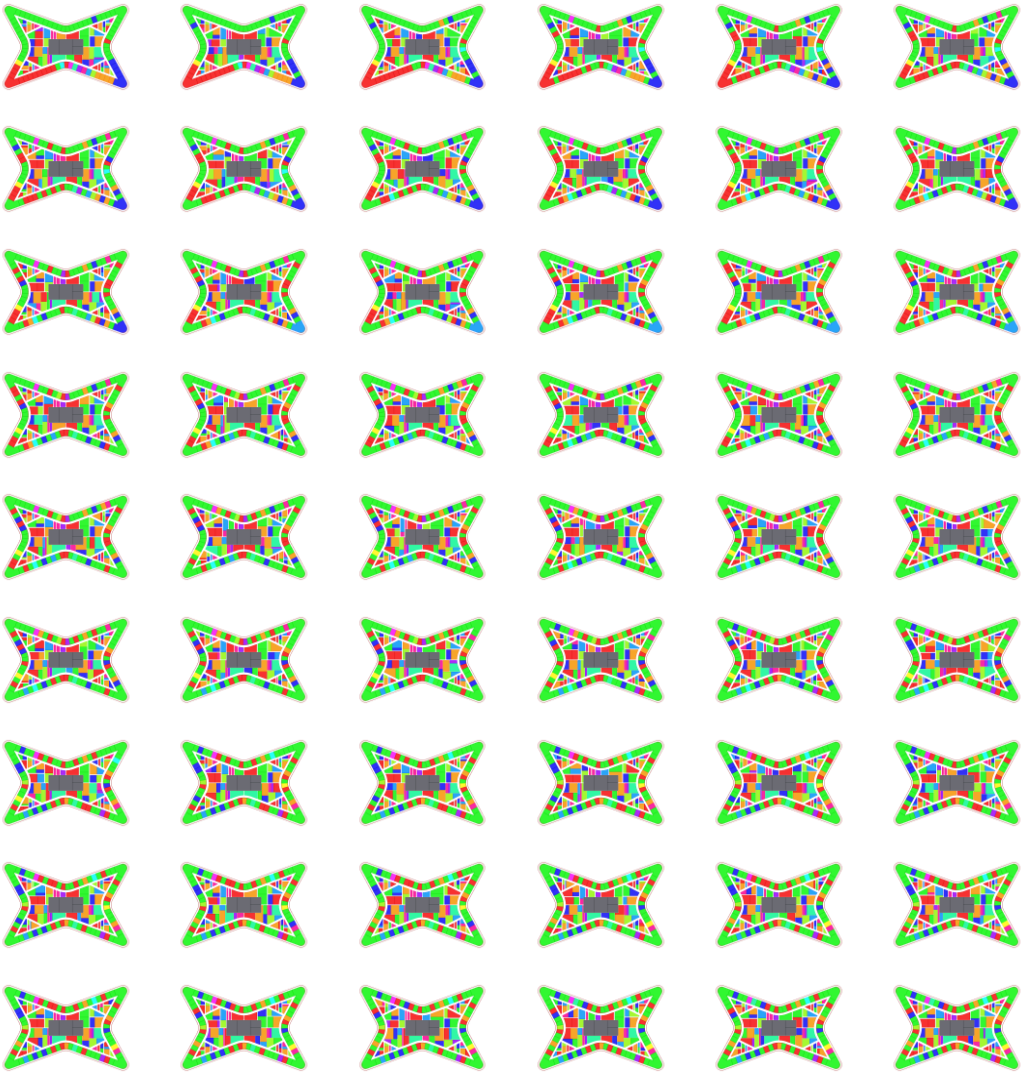
Table 27. User input for spaces in zone-2 in Case Study A

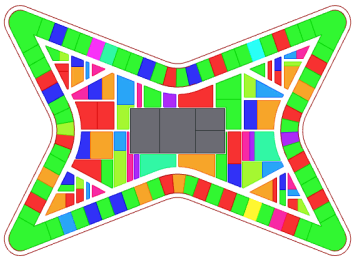
Names	Id	Area	Number	z2.a	z2.b	z2.c	z2.d	z2.e	z2.f	z2.g	z2.h
Workstation	z2.a	1.5	1								-100
share office	z2.b	1	1								-100
public_tlt	z2.c	1	1								-200
IS	z2.d	2	1	100							
dir_office	z2.e	1.5	1								
trash_recycle	z2.f	0.75	1								
Linen	z2.g	0.75	1						100	100	100
Pts	z2.h	0.5	1	-200	-200	-200					
Elec	z2.i	2	1							50	

Table 28. User input for spaces in zone-3 in Case Study A

Names	Id	Area	Number	z3.a	z3.b	z3.c	z3.d	z3.e	z3.f
Trmt	z3.a	1	1		100				-300
Respite	z3.b	1	2	100	100				-300
share office	z3.c	0.75	1			200			
food_gallery	z3.d	1	1					100	100
ptot_therapy	z3.e	0.5	1				100		200
Evs	z3.f	0.5	1				100		200

Table 29. Floor Plan Case Study: Given existing outer form and requirements





Sample of generated variations (Color-coded spaces)

Table 30. Floor Plan Case study- Same workflow as Table 29and updating form.

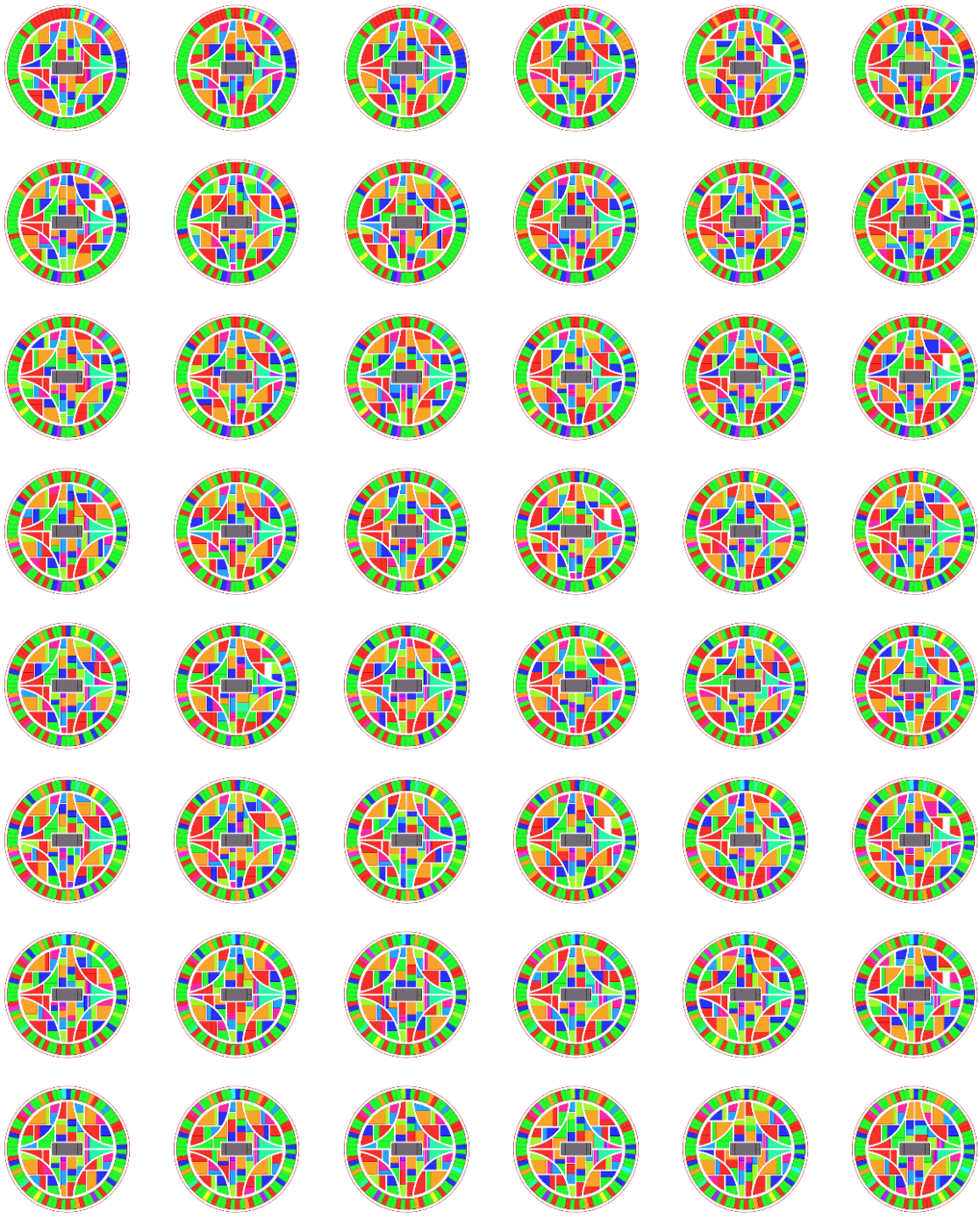


Table 31. Floor Plan Case study - Same workflow as Table-29, 30 with updated forms

5.3 Optimization

(Can a scalable algorithm be developed to resolve many *types* of constraints? Adjacency Matrix, Proximity Matrix, Routing)

The problem of space allocation at each scale (section 1.2) was addressed by specific optimization techniques. The problems required a further classification of design processes to design efficient optimization algorithms. It was found that three types of optimization algorithms were operative:

i. Completely packed regions for space planning are solved using reinforcement learning techniques. The problem requires constant iterations to gauge the efficiency of the layout. This solution is used in space planning and organization of nodes in space-activity networks. They are best-suited to solve the adjacency matrix constraints. When the constraints are provided, they generate a single solution. (Test Case A, B)

ii. Partially occupied regions for site feasibility & parcellation-massing studies. This problem is used in site feasibility studies where the problem is over-constrained. This leads to numerous equivalent solutions. (Case Study B)

iii. Graph algorithms for connectivity in networks of spaces. Apart from organizing the nodes of the network, the space-activity relations require additional solvers to generate the circulation networks. They are solved using the shortest path algorithms using the inputs of cost-constraints. With appropriate constraints, a single-most optimal solution can be determined. Described in detail in section 3.5.

Optimization of activities and geometry is the primary objective of space allocation problems. As described in section 3.2, an abstract model of the layout is used to allocate activities (Figure 41). It is implemented by mapping fields of data structures of spaces to their activities and the relations are used as a graph of nodes and edges. This interpretation of a layout is resolved by optimizing the correlation between arrays of spaces and activities using the techniques based on reinforcement learning (section 3.2). The optimization modules satisfy constraints that are generated from the inputs provided by the user.

The optimization process is developed to support a flexible design process. They are synchronized with the inputs such that it is possible to pause the process, alter the inputs, and resume the optimization process. The illustrations in Tables 32, 33 demonstrate the feature.

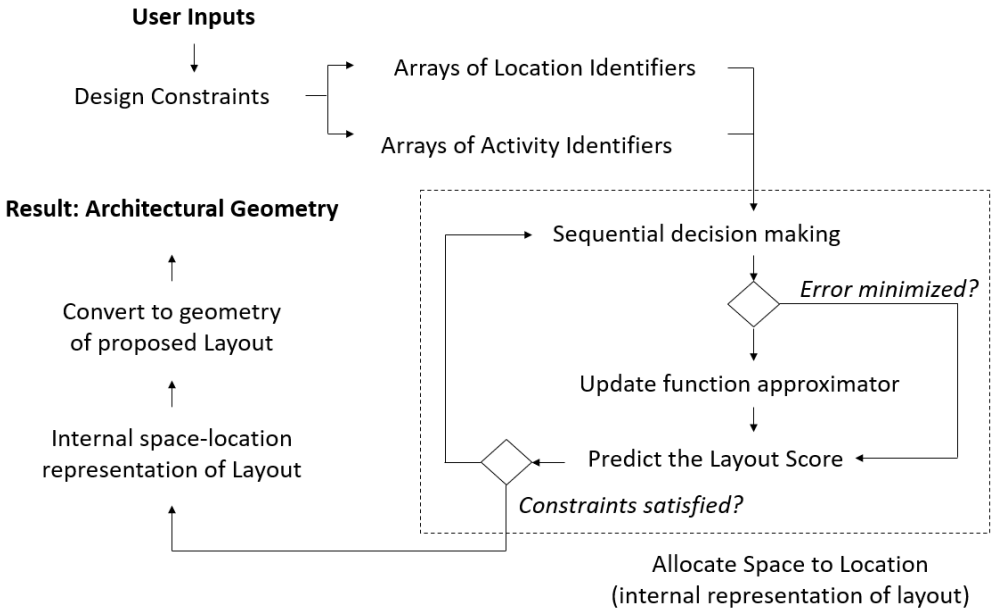


Figure 41. Schematic representation of the optimization process used in this study.

An optimal configuration of the generated layout is the numerical score of the deviation from the adjacency and parametric requirements. In the case of a search process, if the optimal condition is not known, an escape time algorithm is used. The output is generated when the performance of the layout is within an error limit or tolerance value. This generates numerous layouts with equivalent validity. Alternatively, the optimal layout can be determined if it exists. This results in the output of a single layout.

Typical constraints were identified (Table 14, p 120) from prior research and discussions with designers. These relations are sufficient to generate an architecturally appropriate layout. An overall configuration of spaces is evaluated based on numerical inputs provided by the user for each desired relation. Various features were integrated into the algorithms to allow continuous external inputs from the user. The user can intuitively change the constraints by updating the values on a spreadsheet or alter the input geometry and propagate the effects downstream.

5.3.1 Test Case A

This case study demonstrates continuous updates in inputs that are accepted by the optimization process. The process halts and updates the constraints generated from the input. Favorable substructures are not perturbed. Table 32 provides overall geometric constraints. Table 33 demonstrates the steps of user interaction.

Table 32. Inputs required for optimization.

Adjacency Inputs

Name	Key	a	b	c	d	e
name_A	a	100	0	10	0	0
name_B	b	0	10	0	0	10
name_C	c	10	0	0	0	0
name_D	d	0	0	0	-50	0
name_E	e	0	0	0	0	10

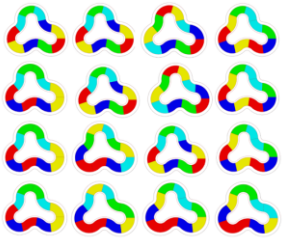
Color Inputs

Key	Color	Names
a		name_A
b		name_B
c		name_C
d		name_D
e		name_E

Geometry Inputs

Name	Key	Area	Number
name_A	a	500	2
name_B	b	500	2
name_C	c	1500	2
name_D	d	750	2
name_E	e	2000	2

Sample Iterations



Step-1 Adjacency Solution

Step-2 Geometry Solution

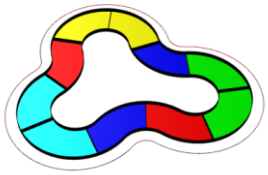
Table 33. Responsive Optimization Process *Halt & Update Layout – 1.*

Specify the Color of Spatial objects					Result
Name	**Key**	**Red**	**Green**	**Blue**	
name_A	a	200	0	10	
name_B	b	0	200	0	
name_C	c	0	0	200	
name_D	d	200	200	0	
name_E	e	0	200	200	

(Table 33 continued)

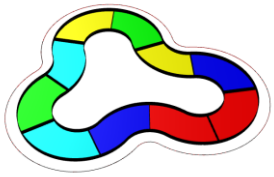
Step 1: Adjacency Matrix — **Layout Solution**

Name	Key	a	b	c	d	e
name_A	a	-100				
name_B	b					
name_C	c					
name_D	d					
name_E	e					



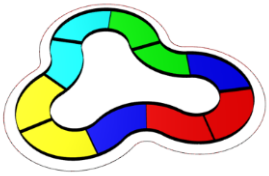
Step 2: Adjacency Matrix — **Layout Solution**

Name	Key	a	b	c	d	e
name_A	a	100		10		
name_B	b					
name_C	c	10				
name_D	d					
name_E	e					



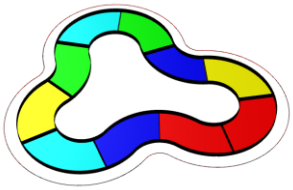
Step 3: Adjacency Matrix — **Layout Solution**

Name	Key	a	b	c	d	e
name_A	a	100		10		
name_B	b		10			
name_C	c	10				
name_D	d					
name_E	e					



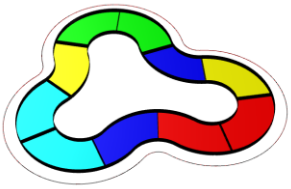
Step 4: Adjacency Matrix — **Layout Solution**

Name	Key	a	b	c	d	e
name_A	a	100		10		
name_B	b		10			
name_C	c	10				
name_D	d				-50	
name_E	e					



Step 5: Adjacency Matrix — **Layout Solution**

Name	Key	a	b	c	d	e
name_A	a	100		10		
name_B	b		10			
name_C	c	10				
name_D	d				-50	
name_E	e					10

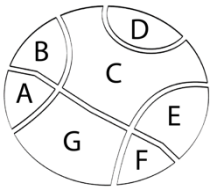
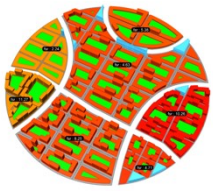
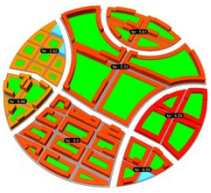
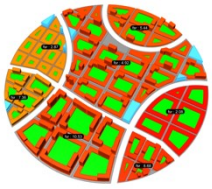
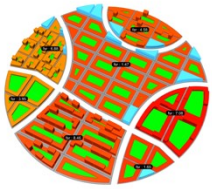
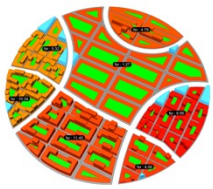
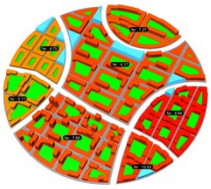
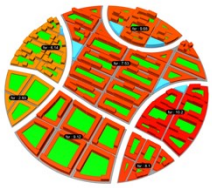
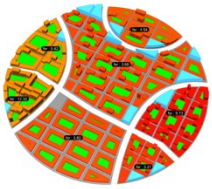


The Table above shows an illustration of the optimization process which can halt for user input and update the solution. If the previous input is changed or requires an update, it is altered. Otherwise, sub-optimal relations are updated.

5.3.2 Test Case B

Table 34 demonstrates the generalization of geometry and achievement of FSR ranges with parcellation and massing modules. Heuristics are used to generate a large number of solutions. In this study, closed and open curves are demonstrated. They may be interpreted as a set of sites or parcels of a site. The gross FSR was restricted while the local FSR was exploratory.

Table 34. Parcellation and Massing

		
Fsr: A, B, C, D, E, F, G	11.27, 2.24, 4.63, 5.38,10.26, 4.71, 8.25	6.94, 2.52, 5.33, 5.61, 6.29, 6.05, 9.9
		
7.38, 2.87, 4.92, 5.44, 2.08, 8.68, 10.53	8.16, 4.33, 1.69, 7.27, 4.19, 5.73, 8.16	5.82, 5.83, 4.99, 4.08, 7.48, 9.0, 7.07
		
3.95, 6.88, 1.47, 4.58, 7.08, 1.65, 8.46	5.11, 4.75, 4.17, 7.27, 8.94, 13.03, 7.68	5.11, 4.75, 4.17, 7.27, 8.94, 13.03, 7.68
		
2.93, 6.14, 7.53, 9.68, 10.2, 8.1, 9.12	4.05, 8.4, 2.31, 5.61, 9.75, 11.94, 7.1	12.32, 5.42, 3.68, 4.54, 5.73, 3.21, 3.82

5.3.3 Case Study B

The site feasibility study provides the program requirements (Table 35) and geometric inputs (Figure 42) for an alternative type of optimization where the area of the site is much greater than the building footprints. Heuristics (genetic algorithms) were used to determine the appropriate locations. The problem is over-constrained and numerous alternative solutions (Figure 43) are possible. From these solutions, the most optimal solution can be retrieved (Table 36, Figure 44).

Table 35. Site Feasibility Studies: Sample of Catalogue & identification of best Solution; Constraints: Table 33

Numerical Inputs							
Tower Type	Number	FSR	Length Min, Max	Width Min, Max	Height	Separation Min, Max	Color
residential	2	0.25	20, 39.5	20, 40	24	10.5, 20	red
low_rise	2	0.5	25, 28	25, 28	54	10.5, 20	green
townhouse	2	0.15	12, 34	12, 34	24	10.5, 20	blue
parking	2	0.25	20, 25	20, 25	36	10.5, 20	yellow

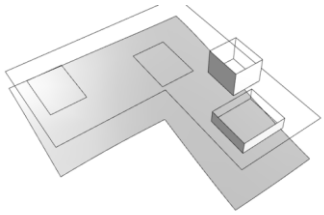


Figure 42. Geometry Inputs: Site curve (plane), sIte topography, excluded regions, and height restrictions.

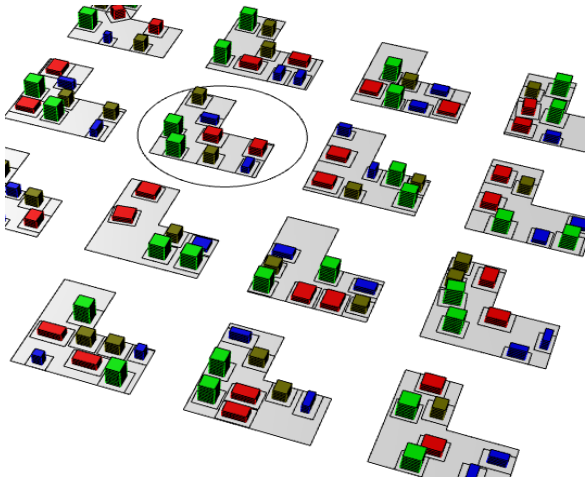


Figure 43. Sample Generated with data exported to spreadsheets above.

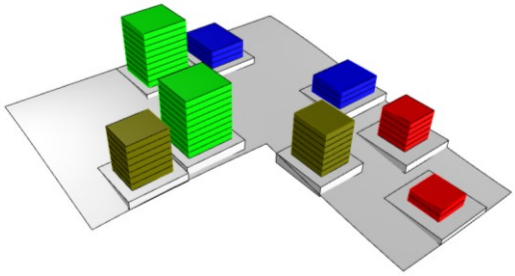


Figure 44. Sample of the generated solution.

Table 36. Output Entry Sample

Type of Building	Number Buildings	Number Plotted	Floorplate Area	Area Achieved
residential_tower	2	14	528	7392
low_rise	2	14	624	17472
townhouse	2	14	240	6720
Parking	2	7	528	7392
Gross Floor Area	38976			
F.S.R	2.996			
Ground Coverage	11.3716			

5.4 Input Interface

The organization of information is addressed by standardizing the types of inputs and the format accepted by the algorithm. Based on the types of constraints (section 5.1), appropriate formats were developed to parse the information. It was found that the following constraints (formats) could be intuitively used in the design process:

i. Correlation between spaces (CSV file format): A matrix where the correlation was specified by rows and columns. For instance adjacency matrix or distance matrix. These are typically topological constraints.
ii. Attributes of the spaces (CSV file format): A matrix that could be parsed row-wise. Each element of the matrix corresponds to the property type. These were used to generate geometric attributes of spaces.
iii. Exposed variables in procedural constraints where the variable could be manipulated directly from the GUI. These variables were linked to procedural constraints.

The proposed models accept information from user inputs (Table 14, p 120). The information represents project-program requirements, bylaws, and design specifications. Inputs from the user are parsed by data processing modules and incorporated into data-structures that form computational constraints (section 4.3.2). The data is transmitted to geometry and optimization algorithms (Figure 45).

This feature allows users to modify intuitive formats and the information is converted into an appropriate mathematical form of the objective function for the optimization process. This allows seamless interaction between the user and the model.

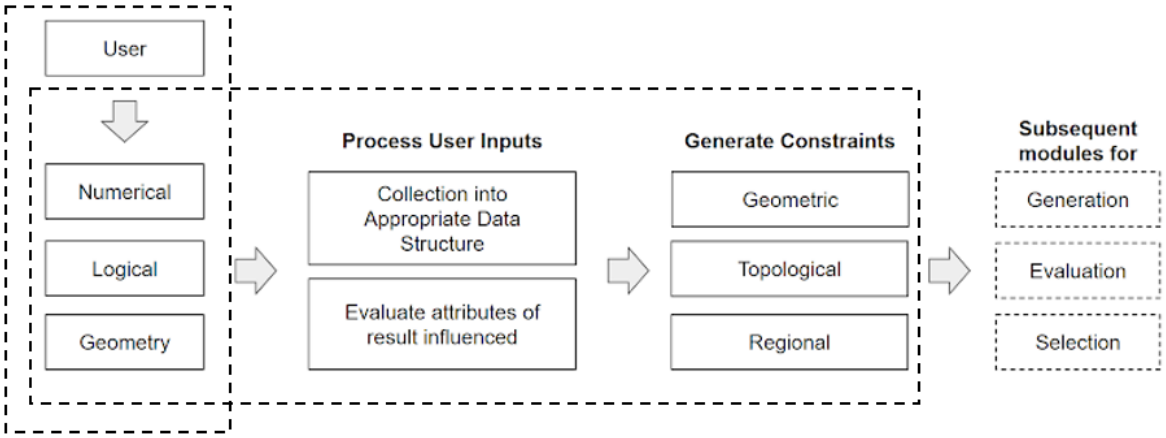


Figure 45. Input interface processes user inputs into constraints.

The data processing modules support *generalization* and *modularization* of the framework (Figure 49, p 175). They transmit information (data) that guide a variety of architecturally appropriate geometric solutions for layouts ad massing. The framework binds the geometry and optimization modules across the design processes. These constraints are available to subsequent processes when the models are connected to emulate

The input interface provides a layer of interaction between the user and the algorithms inside the models. The constraints generated by the interface are large sets of equations that must be satisfied by the optimization algorithms. The generation of these constraints is non-trivial and requires specialization. The standardization of inputs and the

generation of constraints from intuitive forms is a critical aspect of computational models. It allows designers to interact freely with the models. This provides a flexibility that leads to meaningful design exploration. Since the models have been developed to consistently operate on constraints, the interface allows users to test the output that is generated as a result of inputs from various sources or hypothetical inputs when data is not available.

5.5 User interaction

(User changes objectives at runtime without loss of information)

The design of a scheme is a flexible process where the inputs and forms are continuously altered. User interaction is addressed by devising optimization algorithms where favorable substructures are identified and remain unperturbed over iterations and updates in user-inputs. Topological structures are used where an internal graph with correlations between nodes and spaces represent the solution. The geometry is generated when the optimization process is completed. This representation allows algorithms to pause and resume without loss of information. This internal model is sufficiently resilient and permits changes in the input geometry.

Enhanced *design exploration* is addressed in this proposal by providing interactive features that allow the user to manipulate the optimization algorithms during execution. During the process of design, several options are generated and evaluated in rapid succession (section 1.2). During this exploration, inputs provided by the user are not

constant. The inputs rely on an interpretation of design theory, analytical study, or gaps in bylaws. In contrast with prior research, this fluctuation of inputs is reciprocated by the proposed algorithms for SAP.

The constantly changing values require an interpreter (section 5.4) to parse and generate constraints. This will update the constraints based on the inputs. The algorithm employs learning methods (sections 3.1 and 5.3.3) which allows small updates without having to recompute the entire solution. The optimization approach (section 5.3) for elaborate layouts or network optimization responds to small changes because they rely on sequential structures of data compared to heuristics such as genetic algorithms. This will permit pause/update without loss of information.

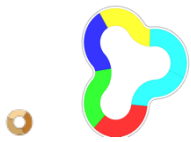
In this proposal, multiple types of generative modules are proposed. Procedural modules can be directly controlled in real-time. Since the proposal allows linked processes (section 4.3), an update to a computational component propagates downstream. It does not require a complete calculation. This is a responsive approach which will enhance the user interaction. This is illustrated by a case study C for site plan development at an urban design scale (section 5.5.2). In this study circulation routes and parcellation were controlled by the user and the floor plan allocation was achieved by SAT.

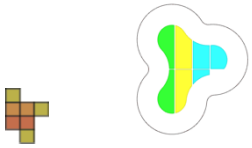
5.5.1 Test Case A (continued) - halt and update

This test case A was introduced in section 5.2.1 where the spaces from a geometric process were perturbed and updated. Using this feature, optimization modules simultaneously operate on numerous geometric processes. The user may continue to make changes to numerous processes while the spaces are configured appropriately. This is an instance of the *halt and update* feature proposed in the thesis. The halt and update feature searches the topological substructure of the layout before updating a collection. Essentially, it queries the local neighborhood of a node for evaluation (Table 37).

Table 37. Responsive Optimization Process *Halt & Update Layout - 2.*

The geometry of Spaces generated separately and optimized by a common solver:

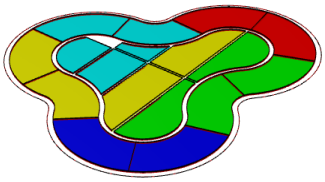




Geometry Setup

Name	Key	Red	Green	Blue	Result
name_A	a	200	0	0	
name_B	b	0	200	0	
name_C	c	0	0	200	
name_D	d	200	200	0	
name_E	e	0	200	200	

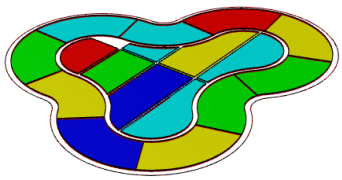
Initial Layout



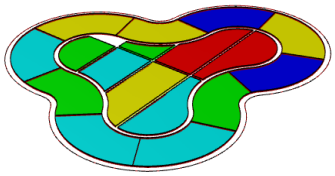
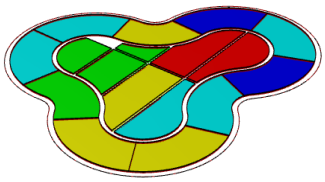
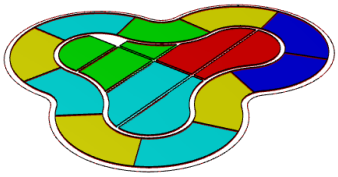
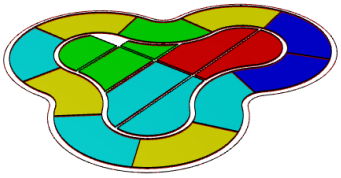
Step 1: Adjacency Matrix

Name	Key	a	b	c	d	e
name_A	a	-100				
name_B	b					
name_C	c					
name_D	d					
name_E	e					

Layout Solution



(Table 37 continued)

Step 2: Adjacency Matrix							Layout Solution
Name	Key	a	b	c	d	e	
name_A	a	100		20			
name_B	b						
name_C	c	20					
name_D	d						
name_E	e						
Step 3: Adjacency Matrix							Layout Solution
Name	Key	a	b	c	d	e	
name_A	a	100		20			
name_B	b		30				
name_C	c	20					
name_D	d						
name_E	e						
Step 4: Adjacency Matrix							Layout Solution
Name	Key	a	b	c	d	e	
name_A	a	100		20			
name_B	b		30				
name_C	c	20					
name_D	d				-50		
name_E	e						
Step 5: Adjacency Matrix							Layout Solution
Name	Key	a	b	c	d	e	
name_A	a	100		20			
name_B	b		30				
name_C	c	20					
name_D	d				-75		
name_E	e					50	

(The optimization process has been designed to respond to the user. The complex geometry of layouts can be generated by multiple components. The adjacency solver operates on the geometry of spaces regardless of the method of generation.)

5.5.2 Case Study C

This case study is a planning project for a science park. It was developed under the guidance of planners at an international design firm. The computational model offered as a solution was based on the site planning models discussed in section 4.2.2. The program requirements (Tables 38-40) included robotic manufacturing, labs, offices, and amenities. The campus integrates high-tech facilities within a parkland with recreational facilities open to residents. It is served by a circulation system that is hierarchical and includes primary streets, pedestrian/cycle street, secondary street, service street, and pedestrian path.

Table 38. Specifications for generic spaces in building-types

Name	Key	Length (min)	Length (max)	Depth (min)	Depth (max)	Perimeter (max)	Height (max)
Manufacturing-I	a	80	120	40	80	500	12
Manufacturing-II	b	60	80	20	40	300	14
HUB	c	130	130	70	70	360	16
Lab/office	d	40	80	20	40	300	16
Amenity	e	20	40	20	40	200	16

Table 39. Specifications for footprints of types of buildings

Name	Key	Number	Footprint Area (SQM)	FAR
Manufacturing-I	a	19	74317.1208	0.44
Manufacturing-II	b	25	56264.1588	0.53
HUB	c	2	14805.1756	0.5
Lab/office	d	4	16489.0793	0.75
Amenity	e	6	6370.16523	0.5

Table 40. Adjacency Matrix

Name	Key	A	B	C	D	E
Low density manuf.	a	100	10	5	5	10
Medium-density manuf.	b		100	40	40	50
HUB	c			100	50	50
Lab/office	d				100	60
Amenity Building	e					-100

The designer can follow the following steps to generate the solution:

1 Primary Circulation corridors for vehicular access to the site

2 The site is partitioned by the circulation corridor

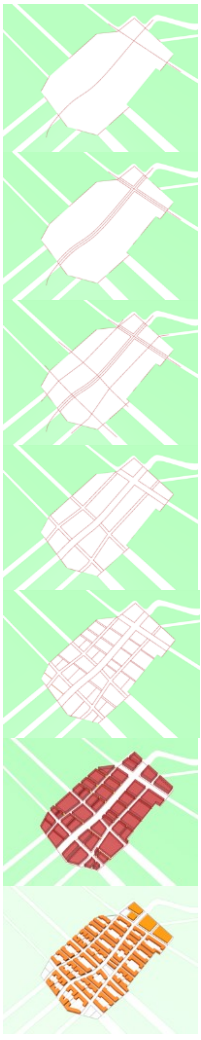
3 Additional hierarchies of circulation are placed

4 The site is partitioned into smaller parcels to host buildings

5 Building block configurations are generated

6 The massing and floor plates are generated

7 Alternatively shaped building formations are developed



Once the floor plates are generated, activities can be allocated to the floors based on adjacencies and area requirements (Tables 38-40). To demonstrate the IDF two types of solutions are provided in figures 46 and 47.

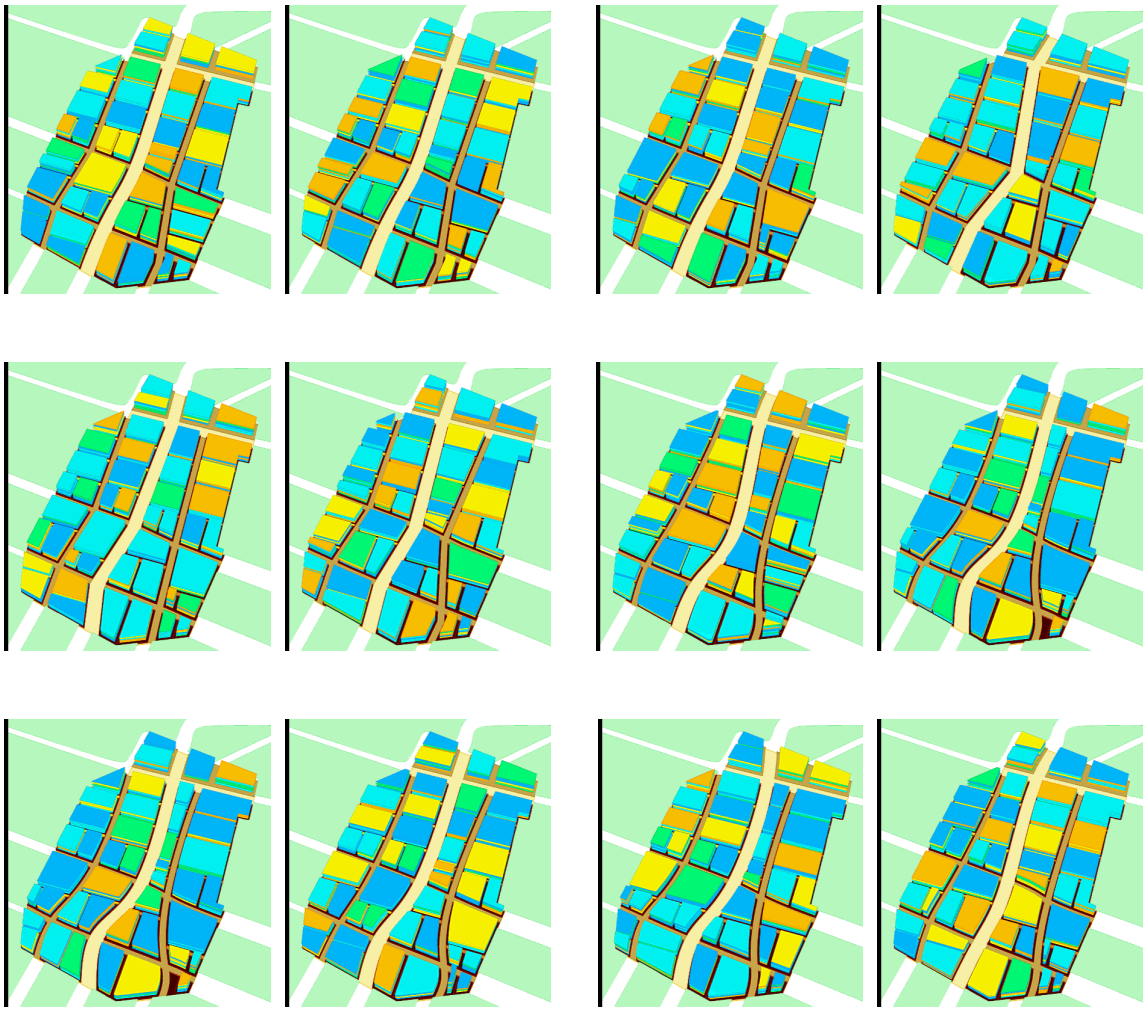


Figure 46. Sample of solutions generated for activity allocation.

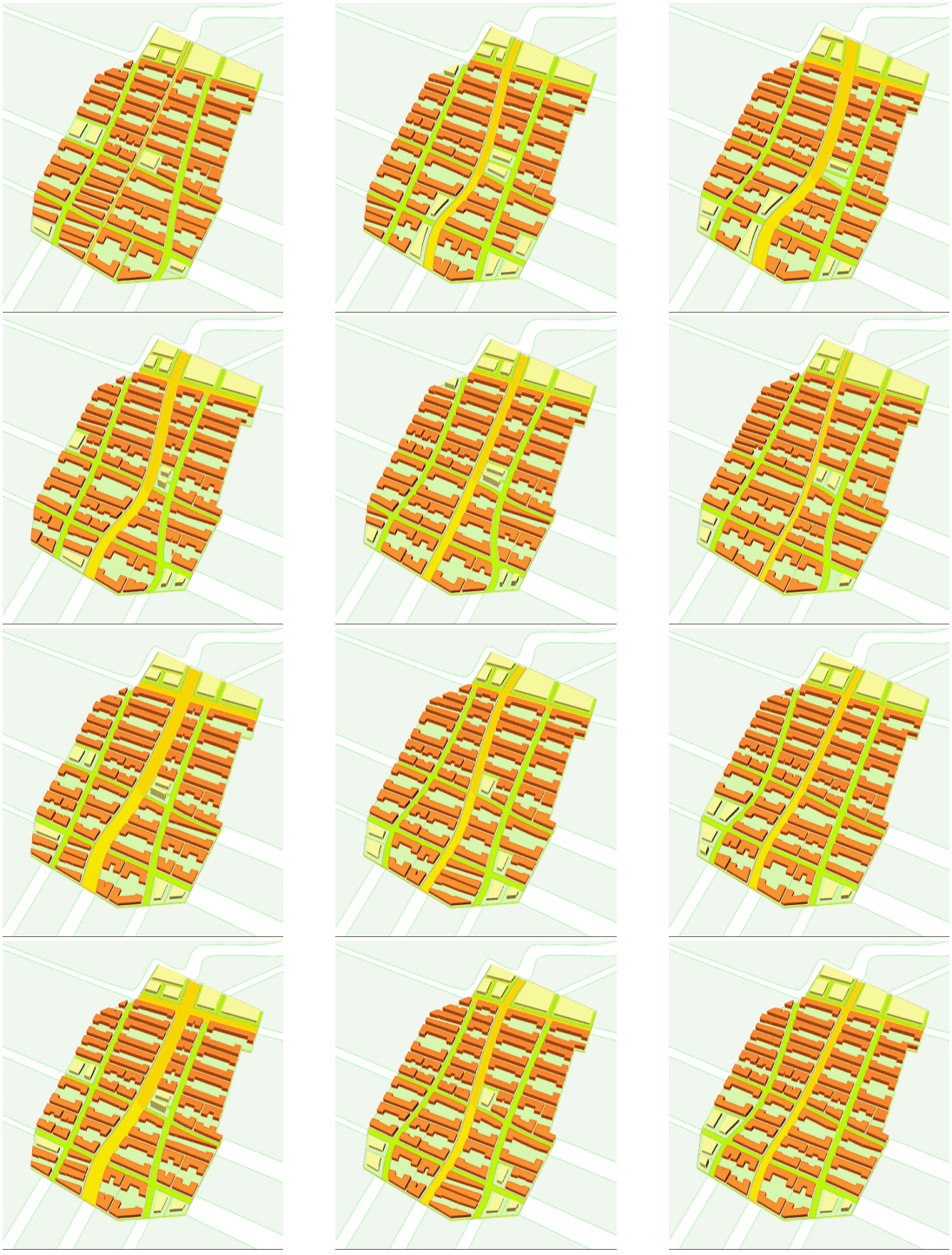


Figure 47. Sample of solutions generated for activity allocation.

5.6 Applications / scaling

The experimental models implemented in IDF provide components that can be used in various permutations (section 4.5) to solve the objectives of the thesis, (section 1.2, p 4) and various permutations of design problems involving the scale, form, and topological requirements. The application of the solutions to the three problems is illustrated in Figure 48. The features proposed in the preceding sections (sections 5.1 – 5.5) permitted the application of the prototypes. In this research, the prototype *IDF* showed that computational methods coexist as layers of information and abstract topological structures inside the system. Interfaces pass the information between users and algorithms (Figure 48 b) and provide seamless processing.

PLUGS-web was a precursor to IDF demonstrates a full-stack web-application for the design and dissemination of the proposal with the explicit intention of participation between designers, consumers, and developers. Using the remote database, it was possible to deploy a large-scale design application where the user can dynamically reconfigure the space-activity relations of the city-blocks of Kyojima district, Tokyo based on the graph-related SAT described in section 3.5 and illustrated by Case study D.

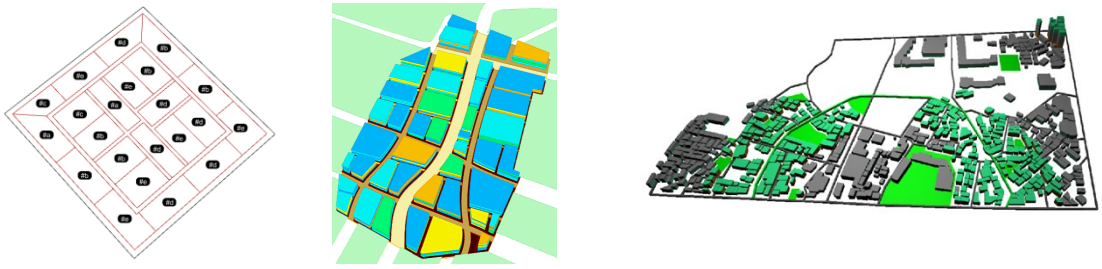


Figure 48 a. Floor plan, site plan, city-blocks

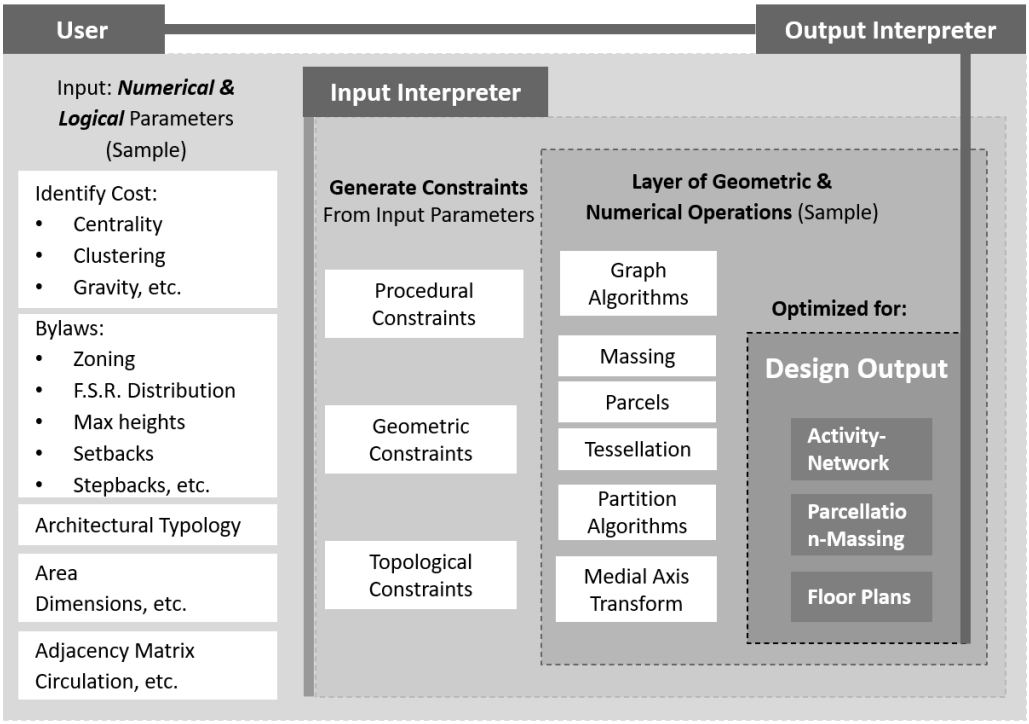


Figure 48 b. Layers of processing.

Figure 48. Applications are embedded in the framework.

The proposed optimization techniques and methods of generating geometric configurations have been extensively tested. Case study A (section 5.2.1) demonstrates the generalization of geometric operations. Test Case A (section 5.3.1) demonstrates the user-interaction with the optimization algorithm at runtime. The halt and update features (section 5.5.1) show the relationship between generation and optimization. Heuristic-based solutions at the site feasibility studies (Case study B section 5.3.3; Case Study C section 5.5.2) plan have been shown. Here, the geometry of spaces generated from various components or even drawn by hand can be used to allocate activities.

IDF provides the ability to explore a wide range of alternatives for diverse design processes and problems. These are features of the computational framework. Based on section 4.4, design exploration is illustrated by:

i. *Independent exploration*: case study A where the initial geometry was altered and reflected in the output generated.
ii. *Substitutive exploration* is demonstrated by case study C where the design decisions are altered while retaining the workflow.
iii. *Alternatives exploration* is demonstrated by case study B where an over-constrained problem generated numerous equivalent solutions.

Design processes in architecture and planning require this flexibility. While prior research used heuristics (section-2) to generate static solutions, *IDF* components allow a dynamic process for design exploration (above). The inputs are intuitive and easily accessible to the designer (section 5.4 and 5.5). IDF components are effective when they are used in conjunction to construct workflows. Each design project can be specifically addressed by customized workflows. This indicates the possibility of a meta-model.

5.6.1 Case Study D

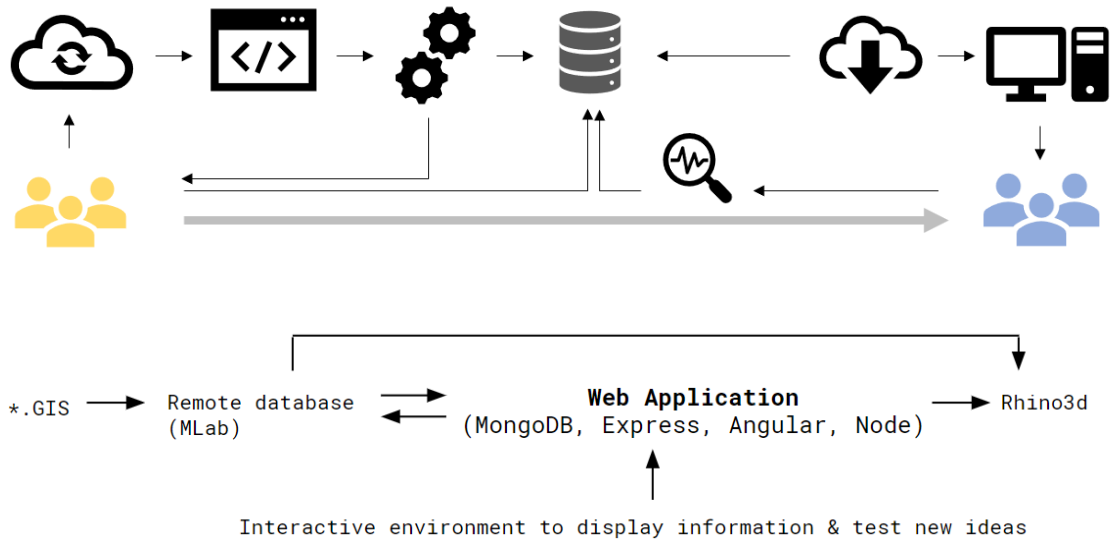
The Plugs-web is a web-based application that runs on the browser. It accepts the GIS input and plots a 3d data-driven environment that can be reconfigured in various ways shown below. There were 2 studies conducted using the data and applying the graph algorithms

to account for urban analysis fields using centrality/dispersal, gravity index, betweenness, etc. These are:

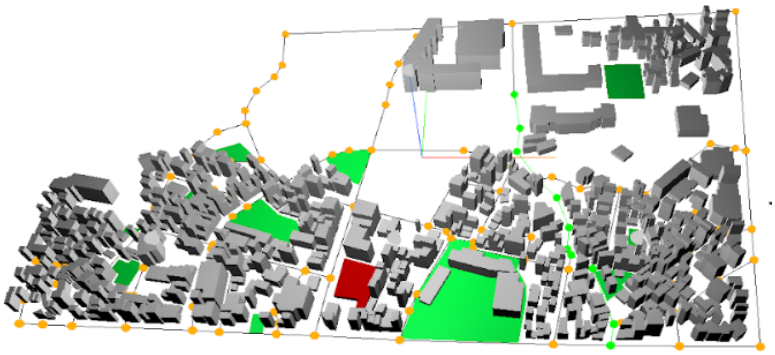
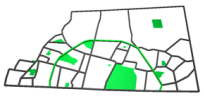
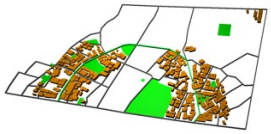
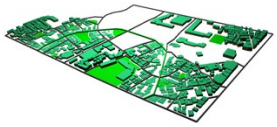
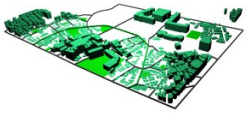
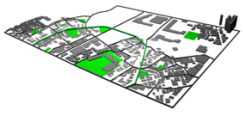
i. Distribution of FAR across all the block boundaries, taken collectively, where the buildings were retained and the type of activity within the buildings was reconfigured based on the allocation of FAR.

ii. Reconfiguration of the buildings within the blocks based on the distribution of FSR. Tessellation algorithms were used to change the built form entirely. Various measures were used. These were implemented directly using a GUI (dat.GUI) framework and allowed designers to alter the inputs at runtime and regenerate the solution.

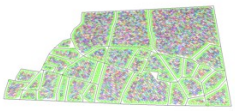
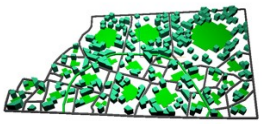
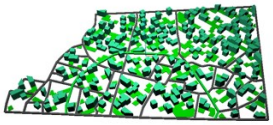
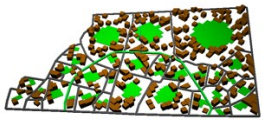
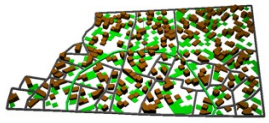
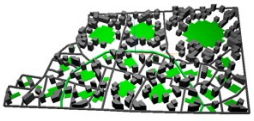
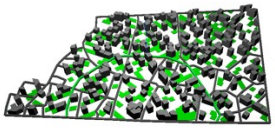
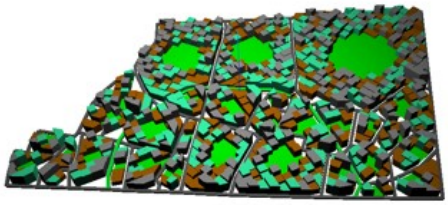
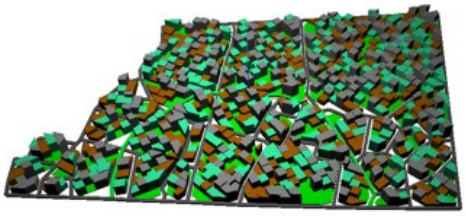
Table 41. Plugs-web for large scale optimization and reconfiguration.



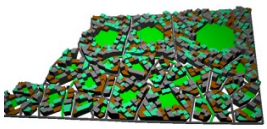
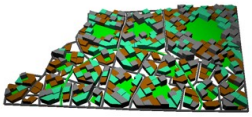
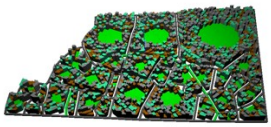
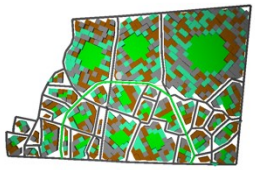
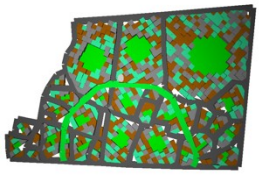
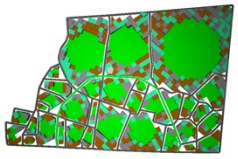
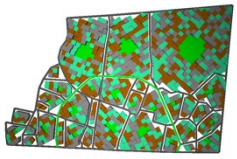
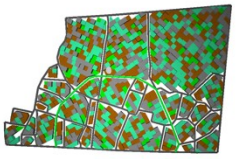
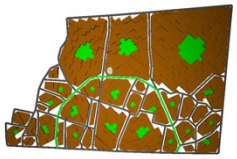
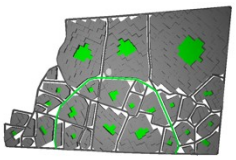
(Table 41 continued)

<table>
<tr><td> 
Streets
 
Parks
 
Buildings</td><td> 
Complete Model
*Kyojima, Tokyo</td></tr>
<tr><td colspan="2">Actual built environment of Kyojima from GIS to browser via remote database (mLab)</td></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>streets</td><td>parks</td><td>buildings</td><td>Complete model</td></tr>
<tr><td colspan="4">Study 1: Distribution of activities (FSR) from the central pedestrian zone</td></tr>
<tr><td>Sample 1</td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>Sampe 2</td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>Sample 3</td><td>  </td><td>  </td><td>  </td></tr>
</table>

(Table 41 continued)

Study 2: Reconfiguration study			
Tessellation			
High Centrality of open space		Dispersal of open space	
(Clustering, fsr, centrality/ dispersal) - Display only GCN & parks		(Clustering, fsr, centrality/ dispersal) - Display only GCN & parks	
(Clustering, fsr, centrality/ dispersal) - Display only NCN & parks		(Clustering, fsr, centrality/ dispersal) - Display only NCN & parks	
(Clustering, fsr, centrality/ dispersal) - Display only RCN and parks		(Clustering, fsr, centrality/ dispersal) - Display only RCN and parks	
Complete Model 		Complete Model 	

(Table 41 continued)

Study 3: Reconfiguration study with updated properties			
Dimension of buildings			
Road depth			
Frequency of element type: park / open space (centralized-decentralized)			
Frequency of element type: GCN, NCN, RCN (centralized)			

5.7 Semi-Automation framework:

The assumption that three different design processes can be linked (section 1.3 and section 4.6) led to the integration of models (components of IDF) by standardizing data structures and correlating input/output fields of components to organize the framework (Figure 49). This permits an exchange of information between modules (Figure 48 b). For instance, the geometry modules of the same type can be used interchangeably.

Alternatively, the optimization modules can operate on any geometric pattern regardless of shape or method of creation.

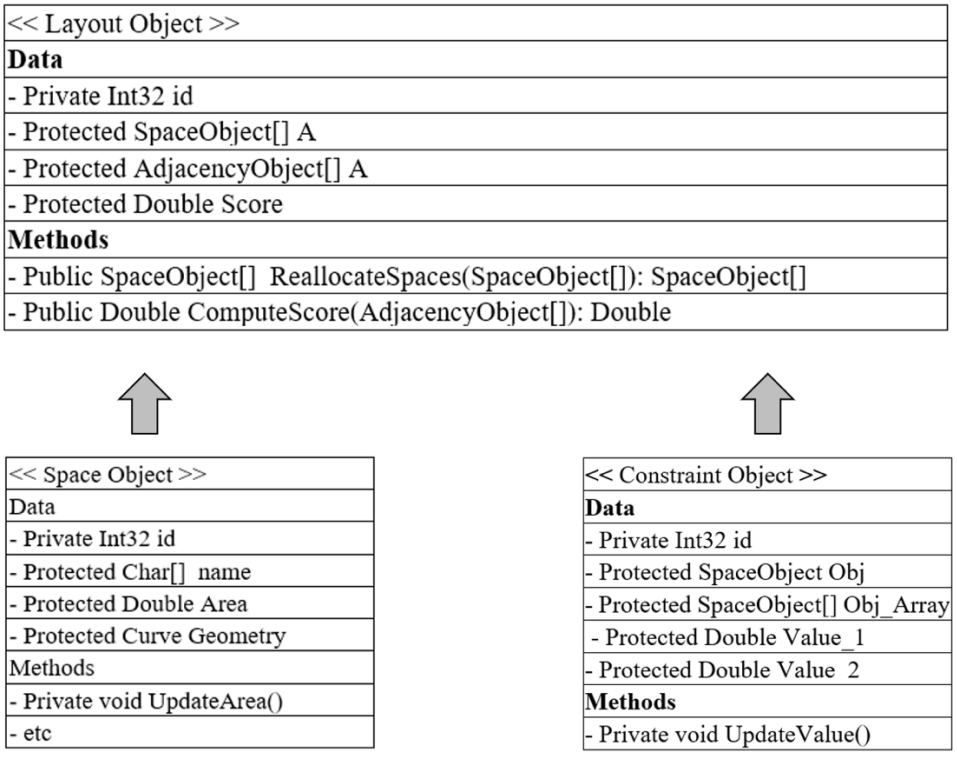


Figure 49. Class structure for an abstract representation of the layout object.

The salient features of the proposed computational framework are illustrated by test cases and case studies (section 5.7.1) and concisely stated as:

i. The proposed techniques to address general solutions to the SAP share common data structures, geometric, and optimization methods. This is achieved by object-oriented programming (OOP) which includes classes, inheritance, polymorphism, etc. (Nakov et. al., 2013).

ii. The organization of the input-output of the components permits successive processing of the connected components across the scales regardless of the type of geometry or constraints. (Figure 48)

iii. The unusual geometric shapes and organization of spaces were addressed by the generalizing geometric operations used in the SAT (section 5.2). The proposed framework is open and allows new modules to be easily integrated using the same data structure and introduce the desired functionality to the framework.

iv. The interactive elements and common data structures (Figure 49) permit the flow of information across the connected modules. This ensures a real-time update. The inputs are intuitive to support designers. The constraints are generated internally from the inputs. This is crucial for applications and generalization of the models. (sections 5.4 and 5.5)

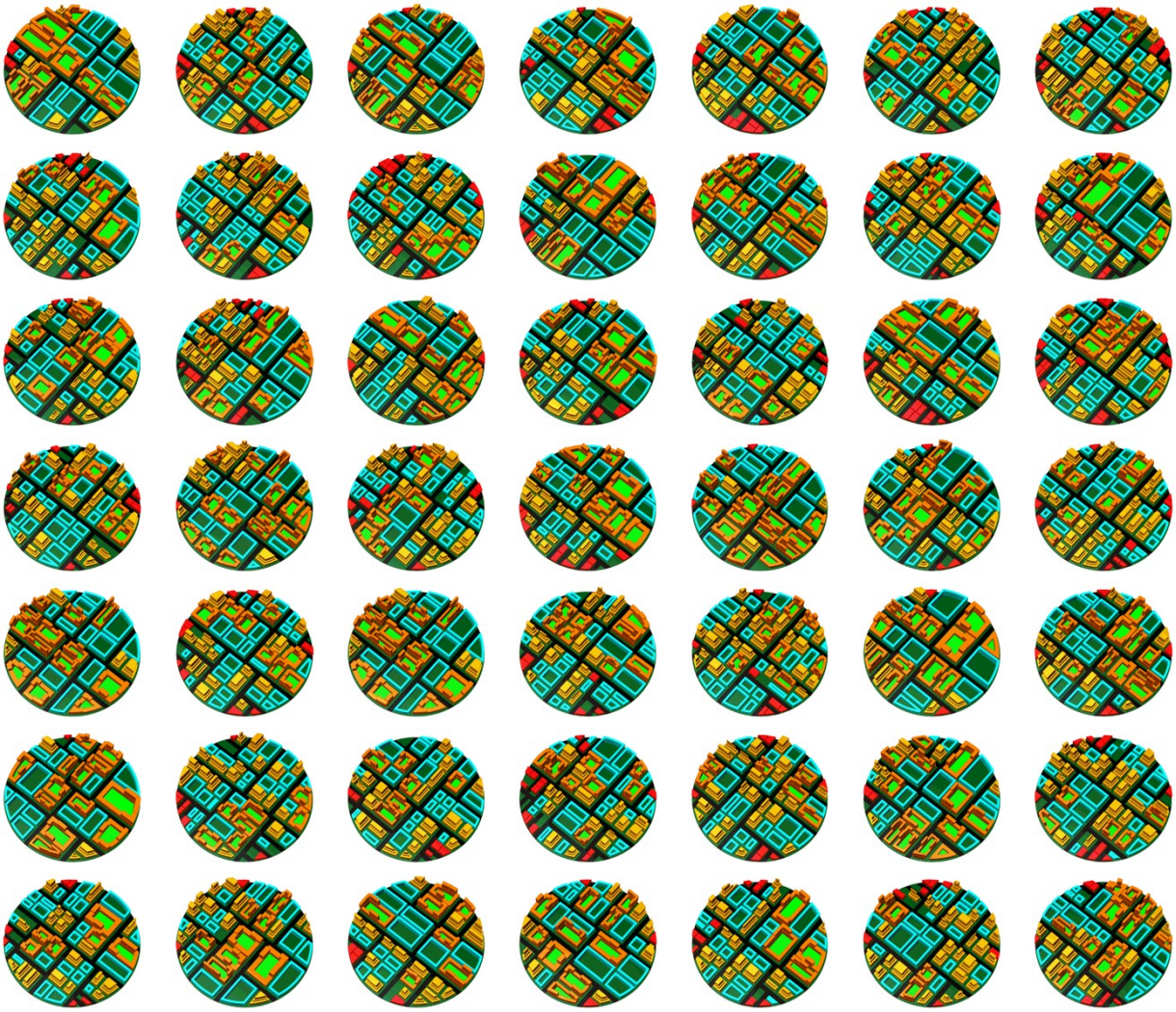
v. The optimization process emulates the flexibility of design processes. Using model-free reinforcement learning techniques, the algorithms can be stopped and restarted without losing information. In a state of pause, the inputs can be changed. When restarted, the optimization module will be able to update the system. (sections 5.3 and 5.5)

vi. Contextualization of problems is addressed by allowing designers to easily generate tentative possibilities across scales (section 5.6). While exact solutions require appropriate inputs, the context allows the visualization of a possible scheme. This is illustrated by test cases C and D in sections 5.7.1 and 5.7.2, respectively.

vii. All the features mentioned above are supported by the optimization and geometric algorithms that lead to the *generalization* of IDF components that implies project-specific *customization.* (Case study A, section 5.2)

5.7.1 Test Case C

Table 42. Comprehensive solutions using mixed typology to generate solutions that meet the requirements for massing, circulation, FSR distribution.



5.8 Hypotheses Revisited

It is proposed that the SAP represents a broad class of design problems in architecture and planning, the solution to which requires a computational approach to combinatorial complexity and geometric intractability. The hypotheses led to the development of several SAT, modules for bundling SAT into typical solvers, integrated data structures and the design of a computational framework. The development of IDF and subsequent applications provided sufficient

5.8.1 A common class of linked design problems

The SAP developed in this thesis is represented by the topological intermediate model, used during the processing of constraints, determines the appropriate layout in that state. The topological representation of the SAP and its existence in various design processes leads to the hypothesis that SAP represents a class of problems encountered in architecture and urban design. The hypothesis is exploited in the design of the computational framework IDF because it reduced the number of modules required for topological optimization.

Using IDF, a design process is interpreted as a series of SAP problems implemented sequentially to generate the spatial output. The models for each SAP are connected by taking the output spaces of the previous problem and using it as an input for the next model as shown in Figure 50. This illustration demonstrates that the application of topological

models for space planning and site planning, where the same component was used with altered inputs (Figure 51). The hypothesis that design processes are linked led to the design of the data structure used in the computational framework such that it allows the user to simply plug-in the models with minimum human intervention.

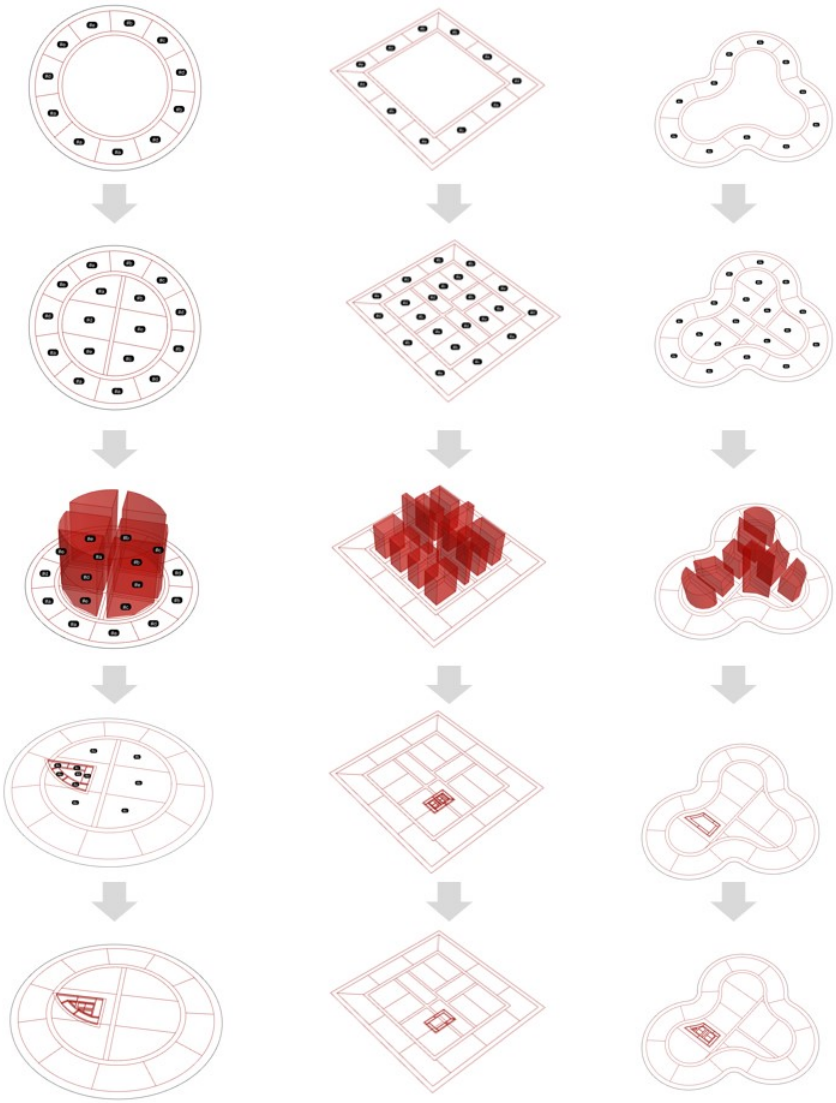


Figure 50. Connecting the IDF components: Parcellation-massing was used to generate parcels, develop the massing. Floors for a mass was extracted. Finally, the floor plan of a building at a level was generated.

5.8.2 Design patterns of geometry and objectives

The SAP-formulation of various design problems is reduced to a graph and optimized. Post-optimization, the graph is transformed back into the geometric form where, procedurally, the appropriate spatial output is generated. The generalization of geometric forms is achieved by operations that not affected by topological variance or the specific scale of the design problem as seen in Figure 51, where the same component is applied.

The illustration presented in Figure 51 also demonstrates that despite the difference in scale of the design process, space planning, and site planning, the SAT of geometric operations and optimization process is common. Their geometric forms may vary based on scale and objectives but there exists at least one equivalent representation of the problems that can be used to solve them efficiently and scale the problem.

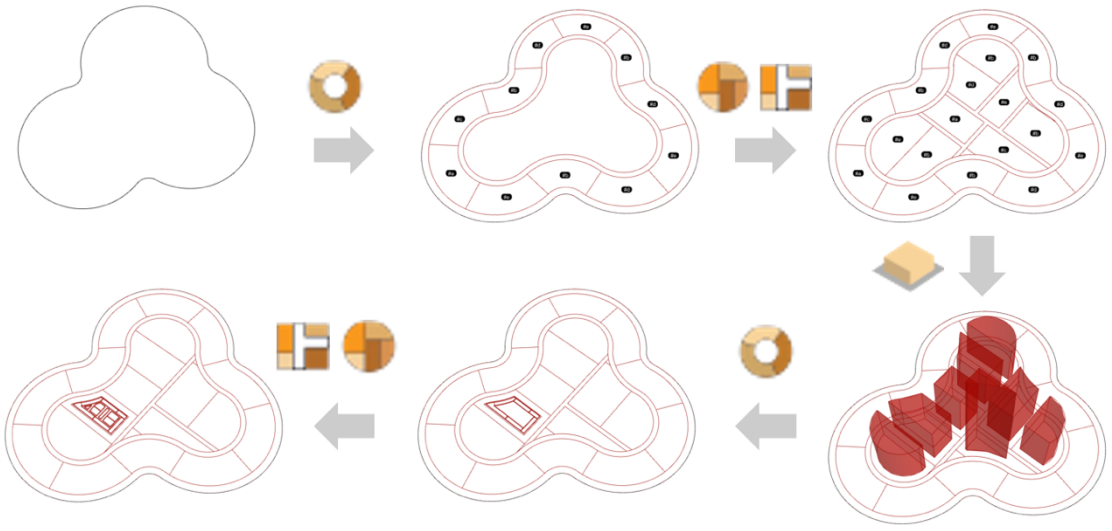


Figure 51 a. A sequence of operations performed by *IDF* components to generate solutions

(Figure 51 continued)

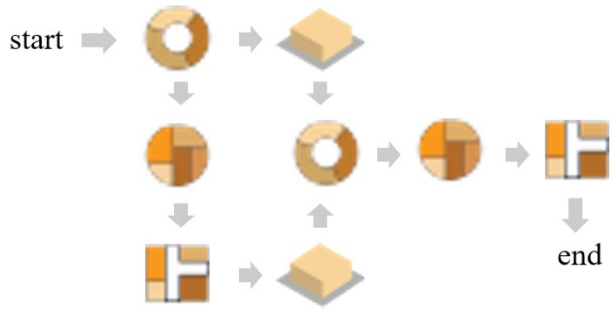


Figure 51 b. IDF notation

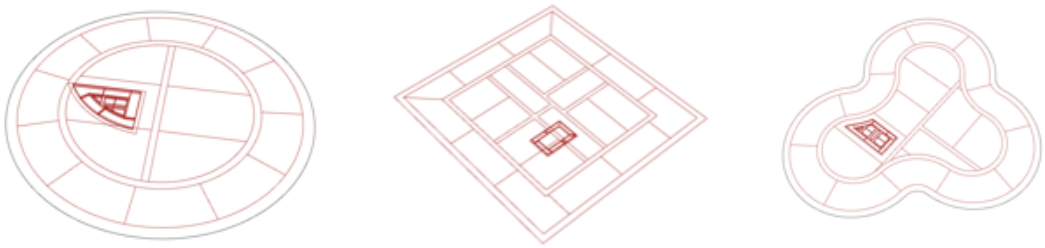


Figure 51 c. Applying the workflow to new design problems

Figure 51. IDF components preserve information about design operations.

5.8.3 Explicit Design Process

The linked generative models or workflows of IDF components reveal features that may potentially lead to the structuring of design processes and design exploration. The proposed SAP models provide a way to store, retrieve, and update architectural information and knowledge which may have favorable consequences for design processes.

From section 5.7, a design problem with many steps can be described as the design processes or IDF components that are connected to form a workflow that approximates the

spatial output of complex design problems (Figure 51). The entire design process shown in Figure 51, consists of site parcellation and massing, generation of floor plates, and finally, space planning can be represented by the components Figure 51 b. For a new design problem, the workflow can be applied with updated inputs to generate project-specific solutions shown in Figure 51 c. This represents the re-use of the design process itself. Or, the design process can be saved and re-initiated with updated constraints. The IDF components provide an avenue to preserve design processes because the components represent the decisions made during the development of the spatial output and the constraints that feed it are generated from the intuitive format of inputs in spreadsheets.

The preservation of design processes has two major components namely (a) sequence of geometric operations, illustrated by figures 52 and 53, and (b) the preservation of constraints and process-data, illustrated by Table 53. (The symbols help identify the component, extensively documented in chapter 4).

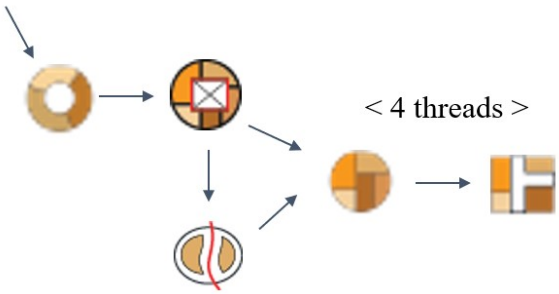


Figure 52 a IDF Workflow

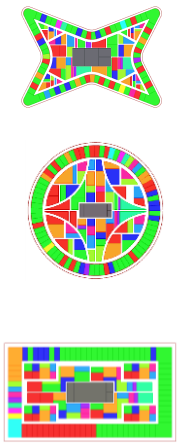


Figure 52 b Application & Re-use

Figure 52. IDF workflow is applied to new forms to illustrate the preservation of structures of design knowledge and information for re-use and evolution (from Case Study A, section 5.2.1).

Table 43. Requirements for Case Study B (shown previously in section 5).

Inputs (Geometry & Adjacency)							
Name	Key	Area	a	b	c	d	e
name_A	a	500	100		10		
name_B	b	500		10			
name_C	c	1500	10				
name_D	d	750				-50	
name_E	e	2000					

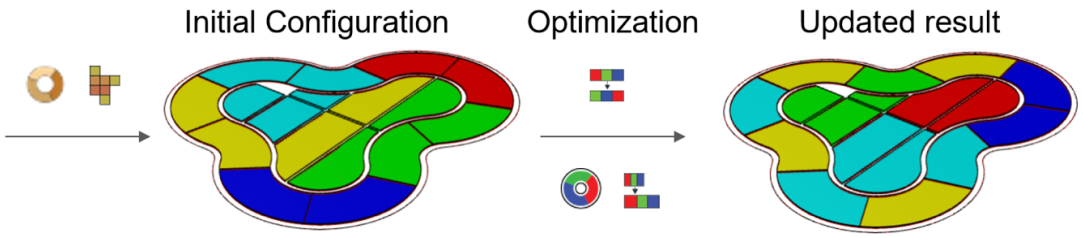


Figure 53. The same optimization algorithm is applied across different ways of generating the geometry of spaces by utilizing information embedded in design processes. Constraints (Table 43) and IDF notation for the Case study are shown.

IDF components or workflows of spatial models contribute to the dissemination of design knowledge and novel problem-solving approach in the following ways:

i. Capture design decisions: The models are representative of the design decisions where a set of geometric configurations are preferred compared to other possible options. The process of decision-making can be recreated by reviewing the application file that contains the workflow of connected models.

ii. Analysis and comparison: The inputs that are converted into objective functions and the attributes of the spatial output can be stored, retrieved, and examined. It can

be used as data to review a project based on the desired and achieved attributes of geometric forms. The changes in the initial assumptions or constraints, over the period of design development, can be reviewed. The analysis can be extended for comparison with similar projects using various statistical techniques.

iii. Evolution of Solutions (design exploration): Since the models are not affected by scale or topological variance in the problems, a workflow of connected models can be re-used for a different problem by updating the inputs. The saving and retrieval of a workflow imply that it can be modified and improved over time as similar design projects are developed.

iv. New solutions: The analysis of projects where the components are used will reveal the shortcomings of the existing components and lead to specifications for improvements, which cannot be anticipated without the use of computable models and their subsequent analysis.

5.9 Design exploration

The proposed models (components) of the framework intrinsically support exploration. The components pass attributes through the input-output fields. This flow of information permits a systematic design exploration. In this research, design exploration has been classified into three types. These are:

a) Independent: numerous independent model-workflows for a design problem.

b) Substitution: altering an intermediate component due to design decisions.

c) Alternatives: stochastic variables that generate alternatives for evaluation.

5.9.1 Independent Exploration

The proposed framework of solvers, IDF, allows designers to set up a workflow of components that perform a set of operations in sequence. Each component contains decision-blocks that guide geometric operations. The components are designed to operate on many instances of geometric inputs. By constructing a workflow of components, the designer chooses to generate a certain type of solution. An entirely different set of components may be used to generate an alternative solution (Figure 54 a). If the initial input such as a site boundary is changed, the workflow updates the output. This allows users to explore entirely different scenarios with minimum effort (Figure 54 b).

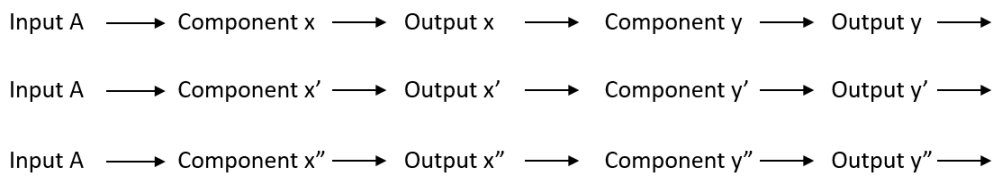


Figure 54 a. Type 1 of Independent exploration where workflows are explored

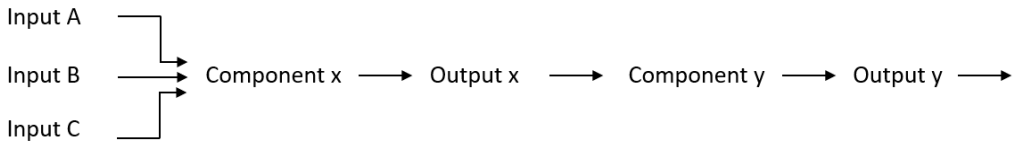


Figure 54 b. Type 2 of Independent exploration where initial inputs are explored

Figure 54. Types of *independent exploration* that allow users to rapidly prototype new design alternatives.

Independent exploration represents alternative solutions to the design problem. Such an exploration is conducted at an early stage of the design process to evaluate overall directions such as constraints or critical geometric forms. Based on the models developed in this research, inductive exploration can be conducted by drastically altering the initial conditions (Figure 55) such as the boundary of the space or constraints. Alternatively, different components may be selected to emulate entirely different solutions.

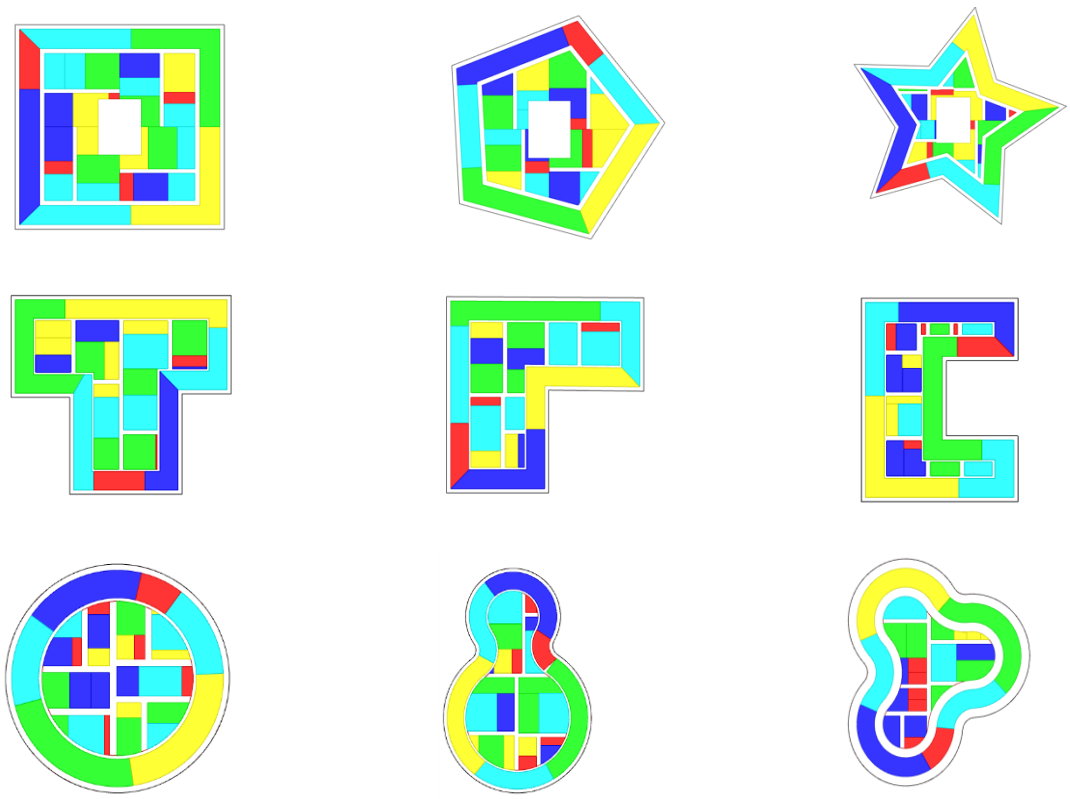
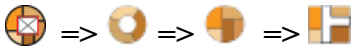


Figure 55. Independent explorations; workflow:  => => =>

5.9.2 Exploration of Substitutes

Quite often, a design problem requires more than one step. This process is replicated by workflows using the proposed IDF framework. Just as architectural elements may be used interchangeably, IDF contains a collection of equivalent components. The designer can explore various solutions by changing the component (Figure 56). This choice is a decision that alters the information sent downstream and influences the result of the workflow.

This aspect of exploration is not representative of the optimization process. Rather, it represents choice presented to the user to utilize predetermined *patterns* such as archetypal partitions of a closed curve and massing typologies. It is an intermediate decision where the remaining workflow is retained and the change in the output is the downstream propagation of the design decision.

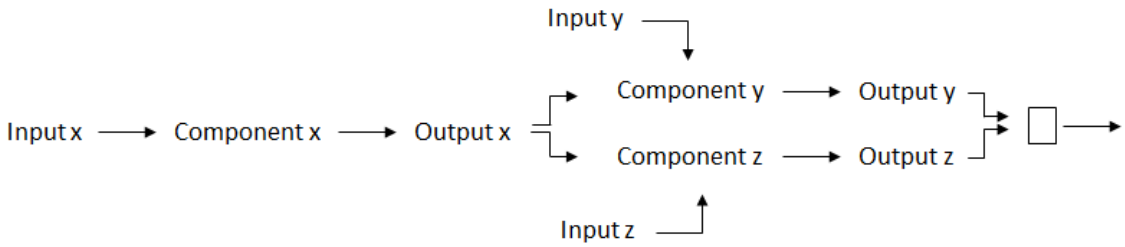


Figure 56. Exploration by substitution: intermediate design decisions.

This represents an intermediate design decision regarding a specific element. The IDF workflows represent a model of the design process. Each model operates or generates

an architectural entity. Components that provide the same class of design solution can be used interchangeably. (Figure 57)

Figure 57. Design decisions to study alternative architectural entities.

5.9.3 Exploration of Alternatives

The process of reconciling user requirements to the spatial output has been developed as an interactive mechanism that generates many variations of the model, where IDF components are models of an architectural entity. This can be facilitated by stochastic variables in the objective function (Figure 58). The stochastic variable may operate within a range, specified by the user. Alternatively, numerous solutions are generated when they meet the objective or the cost function of the optimization process. Over several iterations, this generates the *alternatives* (Figure 59). Or an *exact value* is determined (Table 52) which leads to a single solution or a range of solutions within a tolerance limit. For instance, building footprints on a site or spaces on a floorplate are organized by proximity or adjacency.

Design exploration not only generates alternatives, but it can also be used to study the constraints. The output generated by each component of IDF is consistent with changes

in inputs. By altering the inputs, a user can study the changes in the solution (Case Study A, section 5.5.1) – *halt and update feature*. It allows the user to make small adjustments and observe the effect of constraints. Over several iterations, the designer can systematically explore various configurations and determine an exact set of constraints.

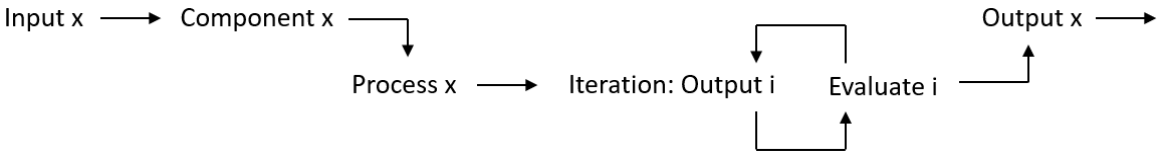


Figure 58. Exploring alternatives.

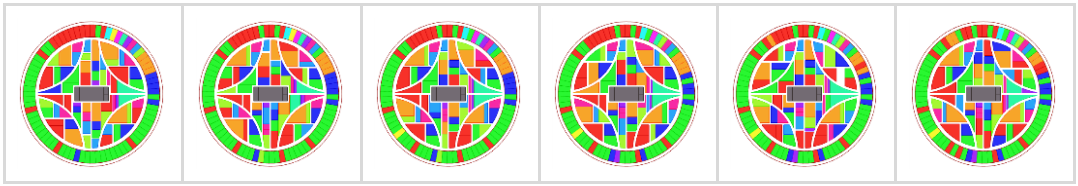
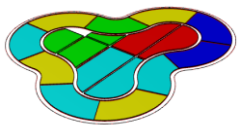


Figure 59. Exploring Alternatives (sample from Table 30, Case study A, section 5.2.1)

Table 44. A constrained exploration to generate optimal solutions– sample from case study B, section 5.3.3.

Name	Key	A	b	c	d	e		
name_A	a	100		20				
name_B	b		30					
name_C	c	20						
name_D	d				-75			
name_E	e					50		

CHAPTER 6. EXTENDED APPLICATIONS

In the previous chapters of the thesis, a generative design framework is proposed which tracks and provides general solutions to SAP across various scales and permutations across scales. Increasingly, as the role of data-analytics informs decision-making, a computational framework such as IDF is essential as a methodology to generate alternative models for evaluation (section 6.1). Among various sources of data, the measurement of energy and performance requirements is a crucial factor in determining appropriate design schemes. Section 6.2 provides a detailed account of generation-evaluation processes, anticipated since March (1976), to develop holistic design solutions that guide practice and analytical research. The solvers proposed in this thesis are compared with the dominant theories in generative design (section 6.3).

6.1 Towards a Data-driven generative processes

Design is not a static process (Liggett, 2000). It is proposed that a computational framework must permit various types of exploration (section 4.4). But the generative mechanism should be such that designers can study the effect of constraints within realistic limits of time and hardware (section 5.5). Manual design practices do not adequately support the integration of analytical tools due to quantitative limitations (Chang et.al., 2019; Yang et. al., 2020). Computational models are necessary to (a) generate alternatives for evaluation and (b) solve constrained problems. In the former, random variables are

exploited to produce a large number of variations in design schemes. These are analyzed by experts to produce design guidelines and constraints.

6.1.1 Available Data & Analytics

Software such as Energy Plus, Radiance, DaySim, Open Studio allows designers to evaluate a given scheme based on performative measures (Anton and Tanase, 2016). Designers can use desktop applications such as Ladybug and Honeybee plugins for Rhino3d (Roudsari et al, 2014) and connect with the aforementioned software to guide design decisions for individual spaces in buildings such as the window-wall ratio, proportions of the sides, fixtures, etc. (Haymaker et. al, 2018). Given a set of buildings and parametric variations for height and orientations, energy analysis, and building simulation studies provide necessary guidelines that improve the performance of buildings or clusters of buildings (Vasanthkumar et. al., 2017; Naboni et. al., 2018). Analytical studies for large-scale planning are fueled by data from various sources such as socio-economic (income-rent) data of neighborhoods, tourist spatial-temporal behavior data, origin-destination travel, sentiment analysis of popular social-media, etc., as well as immersive technologies and realistic simulations of the proposed scheme allow designers to gather feedback (Bagan and Yamagata, 2012; Yamagata et. Al., 2015; Kellner and Egger, 2016; Caldeira and Kastenholz 2019). Smart city paradigms are being envisioned by sensors and IoT devices that provide real-time data to predict usage of the spaces and state of the environment wrt mechanical systems over time to improve the health, energy, transportation, etc. (Al Nuaimi et. Al., 2015;). Apart from this, several cities such as New York, San Francisco,

Los Angeles, etc have open data platforms that are available for analysis and usage. These data sets provide information about various aspects of the city such as healthcare, transport, land use, demographics, etc.

6.1.2 Using the Data

Computational frameworks such as IDF and PLUGS anticipate the need to make design decisions based on data from prior sources and use analytical tools to generate recommendations for project-specific optimization. The user may synthesize data and analytics from the *context* or any other suitable source to develop constraints that can be consumed by the models (Table 6). Various ways in which the proposed models support data-driven solutions:

i. Attraction/repulsion/orientation: This form of interaction between spaces is proposed as the fundamental motivation for organizing spaces. Governed by sequential updates, an agent finds the optimum position for each space to satisfy pair-wise relations between spaces and activities. The attraction and repulsion set up between spaces encode a vast number of desired objectives services such as:
    a. Grouping based on structural load calculation occupancy & equipment. Similarly, HVAC loads, plumbing can lead to grouping or separation.
    b. Separation values govern the exposure to the external face of the building for supplies, daylighting, ventilation, storage, and security, etc.

c. Attraction values govern the central location of spaces which may be used to identify public or private spaces at the core and branch out.

ii. Centrality/Dispersal: The interacting networks of space-activity relations utilize information from travel-destination studies to position nodes and edges such that the networks are optimized in terms of connectivity and proximity between nodes of activity. A node is be considered central to all other nodes or sub-groups, known as cliques which may also be constrained. Not only the nodes but groups of nodes are positioned according to distances from a point or curve – locus.

iii. 3d-Displacement: The mixed-use programs developed by planners utilize socio-economic data and the program guides the mix of spaces and generates the spatial organization required by the design problem. The gross activity allocation or zoning of floor plates of configurations of buildings ensure proximity between buildings and elevation of the floor plate. For instance, a retail network at lower floors followed by offices and residences is a common mixed-use typology employed in towers. The grouping of floor plates for allocating activities also governed by the distance between the floor plates. Essentially, all the floor plates are considered collectively leading to a three-dimensional spatial allocation for groups of buildings.

iv. Constrained Solvers: Floor plan layouts of buildings are organized based on area requirements of spaces, the adjacencies, and orientation. Each of these objectives is supported by prior inquiry. Their association provides sufficient conditions to manipulate the model.

v. Participation & Continuous Process: The proposed computational framework such as IDF and PLUGS allows designers to set up constraints, generate the solution, and reiterate with updated constraints. The memory and time complexities are handled to set up a continuous loop that sequentially updates the scheme and allows the user to pause and update constraints without a loss of information – optimal and sub-optimal substructures of the scheme. This halt-update feature is an essential algorithmic design because constraints are flexible in architectural design and adjustments are necessary during the process.

vi. Linked Spatial Solutions: The proposed models allow data and constraints across various scales because the workflows of connected models generate solutions across numerous problems and scales, etc.

vii. Review/Memory: By examining the schemes generated by a conventional design process – the drawings, it is difficult to discern the design decisions and or the sequence of operations that led to the result. It is anticipated that if the computational framework is used in practice, the workflow will reveal how the solution was implemented simply by examining the components used because each component serves a specific purpose and it encapsulates a series of geometric and logical steps. This information can be used to disseminate the knowledge accrued, use, and evolve the workflow over similar projects or compare similar projects and develop predictions.

6.2 Generation-Evaluation Processes

An objective of this research in generative design is the exploration of the alliance between generative design and performance-based analytical processes to generate recommendations to designers for subsequent design development. Design alternatives were required as inputs to analytical studies in buildings and urban forms. The potential for quantitative solutions was exploited by colleagues who are researching simulation techniques to analyze the alternatives. The measurement of daylight or enery consumption is a computationally intensive task which requires time, in a way that the output cannot be used as feedback mechanism. A large number of design options are generated and connected to EnergyPlus via LadyBug/HoneyBee components in Grasshopper3d software. This analysis of layouts was conducted by professionals at an architectural firm to explore the extent to which generative technologies may influence practice. Two research studies are presented at the urban scale (campus planning) and multi-level buildings to demonstrate the use of proposed generative techniques alongwith energy and performance analysis in order to provide an "environmental solution".

6.2.1 Generate-Evaluate process in Urban Design

The generative problem was addressed as a site feasibility study (section 1.2.2.3) and the models described in section 4.2.2.2 were utilized. Chang et. al. (2019) provides a detailed account of this research in generative design to develop a methodology for a performance-based approach in urban design. A summary has been provided in this thesis

to demonstrate the alliance between generative techniques and analytical research to aid the development of urban design schemes.

A computational model was developed using the techniques proposed in this research to generate numerous configurations of a set of buildings based on proximity, dimensions of buildings, topography, etc. The alternatives were used to study multivariate relationships in campus planning. The design options were analyzed to determine the optimal conditions for solar radiation, energy performance, and sky-view factor. The multivariate analysis demonstrated relationships between urban geometric forms and performance criteria. The findings provided recommendations for i) optimal solar potential and the building coverage ratio ii) the optimal energy balance and the threshold for sky view factor. These relationships contributed to the development of design strategies and guidelines for designing a sustainable campus.

6.2.1.1 Setting up constraints for generation

The site was located in Shenzhen, China (Figure 60), where the land is constrained by low-lying terrain. The east of the land is relatively flat, and the terrain is bumpy to the southwest. The climate was analyzed based on the weather data collected from the Solar Wind Energy Resource Assessment (SWERA) project funded by the United Nations [29], and hourly weather data was collected from a database of 14 developing countries including China. The Shenzhen weather data (.epw file; EnergyPlus Weather) collected from 1980 to 1998 was utilized to understand the climate condition of the site. According

to weather data, the maximum temperature is about 36.7 °C in July and the minimum temperature is about 6.1 °C in February. The monsoon seasons are from May to September, and more than 30,000mm of rainfall on average are poured. The daily solar radiation is estimated at 3.57 kWh/m2/day in February at least and about 4.78 kWh/m2/day in October at most.

The topography is constrained by splitting up the surface of the site into discrete grids and measuring the rate of change of slope. Based on this value, sites are considered as buildable. The minimum and maximum values were parameterized to allow real-time control over the generative mechanisms. The rate of change in the slopes is tracked (Figure 61) to determine the water flow and circulation patterns. This information is extracted from the environment and used to generate solutions with optimal scores.

In addition to the slope of the site, a horizontal pedestal or base (Figure 62) was generated at an optimal level to site the buildings. Project-specific modification to the computational model allowed designers to place the pedestal and the building formations were according to a predefined configuration and manually selected zone. However, the analysis was conducted on layout output by the generative algorithm.

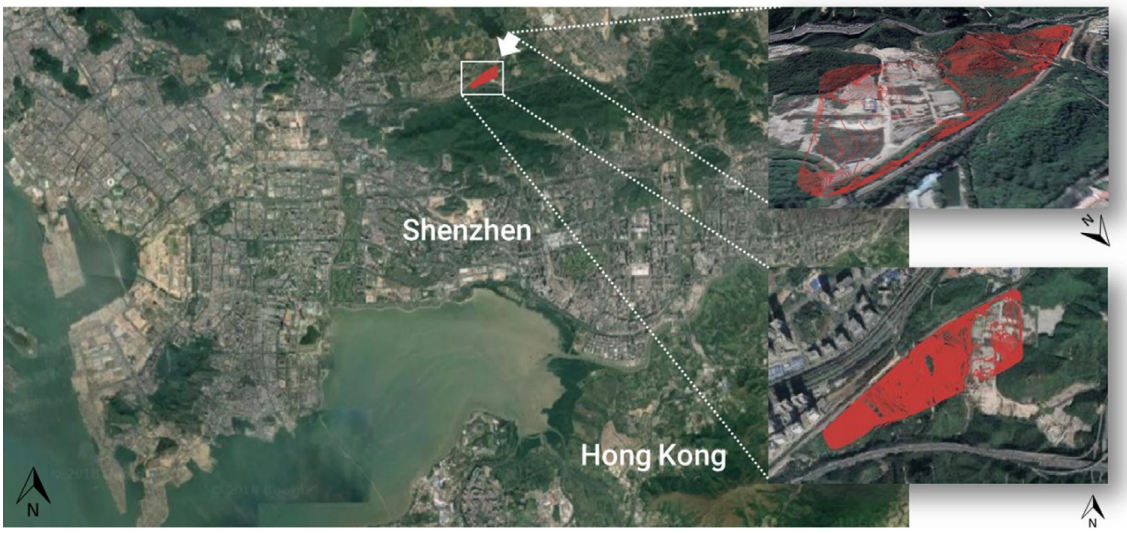


Figure 60. Study Area

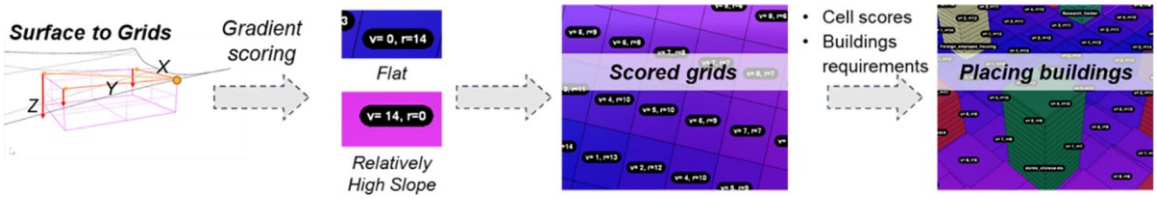


Figure 61. Gradient parcels and building generation for the clustered and decentralization scenarios.

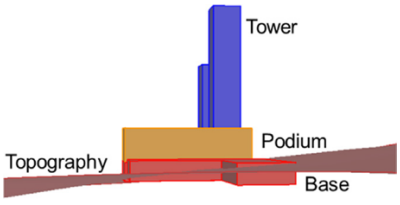
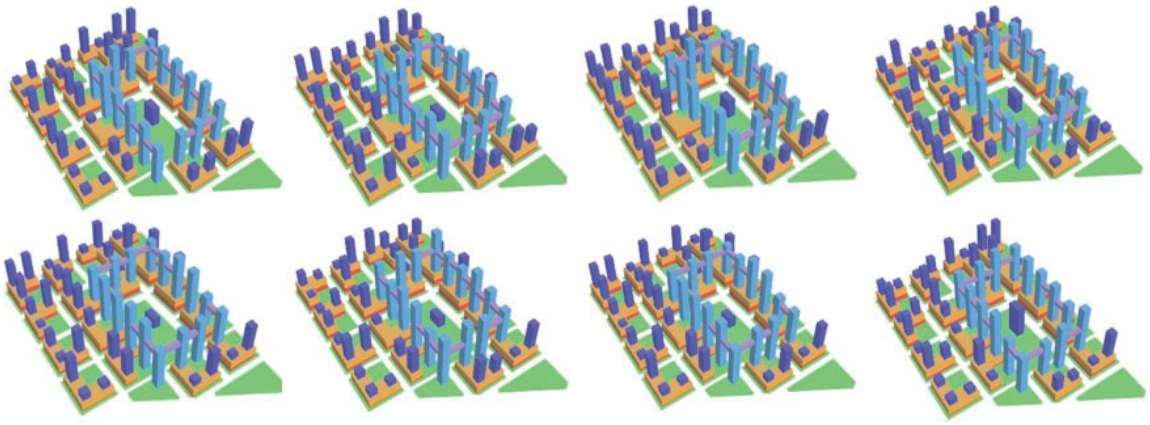


Figure 62. Type of agents in the reinforcement learning for the concentration scenario.

The constraints for the feasibility study is developed from the project requirements (Table 44). The model is described in section 4.2.2.2. The additional step for floor plate allocation was not developed in this study. Due to this reason, a vast number of possibilities emerged (Figure 63). The geometric results of the layout of building placement and massing were used in the subsequent section of the research to ascertain trends for optimal sky view factor, solar heat gain, and energy savings.

Table 45. Constraints from program requirements

Building Type	Area	Min Length	Max Length	Min Width	Max Width	Min Height	Max Height	Min Separation	Max Separation
classrooms	10800	20	25	15	25	12	15	4	7
library	7170	20	25	15	25	13	15	4	7
labs_space	45120	20	25	15	25	23	30	7	10
sports_gym	3420	20	25	15	25	25	30	7	10
Auditorium	1920	20	25	15	25	5	15	7	10
dorms_chinese-bs	14400	20	25	15	25	21	30	7	10
dorms_chinese-ms	12555	20	25	15	25	22	30	7	10
dorms_chinese-phd	2880	20	25	15	25	25	30	7	10
dorms_foreign	17427	20	25	15	25	26	30	7	10
Cafeteria	3900	20	25	15	25	15	30	6	10
auxillary	11250	20	25	15	25	15	30	5	10
uni_admin	5525	20	25	15	25	12	15	1	3
dept_admin	6712	20	25	15	25	14	30	7	10
Faculty_staff_dining_room	1188	20	25	15	25	3	8	8	10
Faculty_staff_housing	19998	20	25	15	25	25	30	7	10
Chinese_employee_housing	7238	20	25	15	25	24	30	9	10
Foreign_employee_housing	11880	20	25	15	25	23	30	4	10
Aux_Foreign_Employee	880	20	25	15	25	3	8	5	10
Ummaried_employee_housing	494	20	25	15	25	3	8	6	10
Research_Center	15200	20	25	15	25	14	30	3	10



(Figure 63 continued)

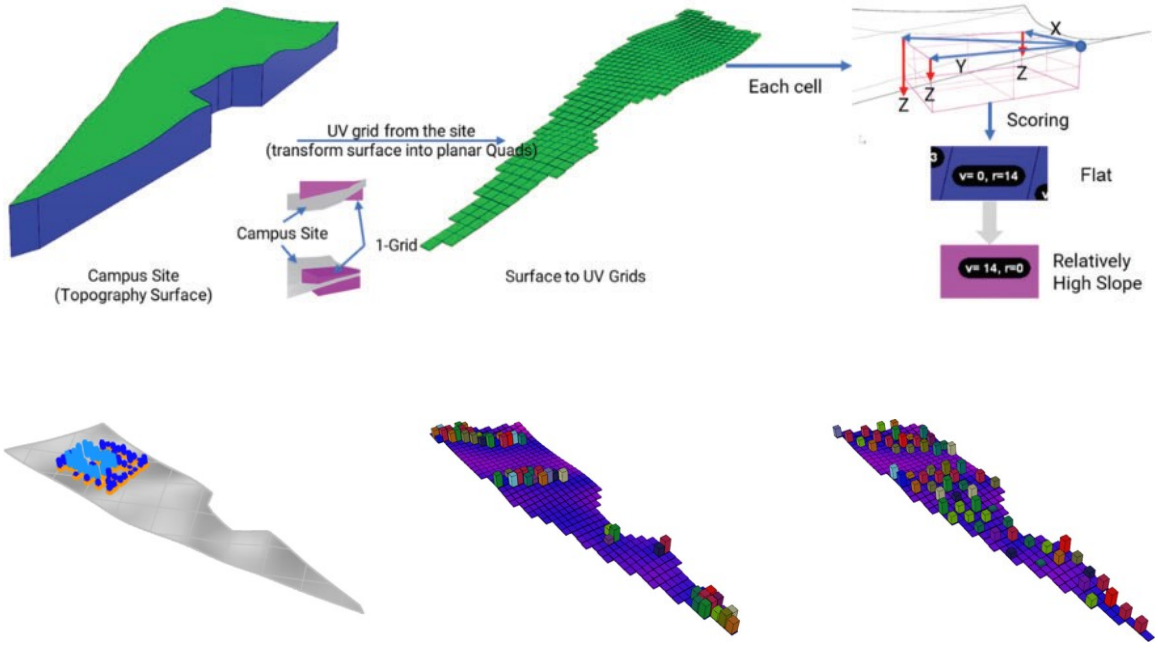


Figure 63. One option out of 10 options for each design scenario: concentration (left), clustering (middle), and decentralization (right).

6.2.1.2 Connecting the generative design to performance analysis

The allied research was conducted to measure the sky view factor, solar harvesting potential, and energy demands. Each scheme produced by the generative apparatus was saved and plugged into the analytical pipeline as shown in Figure 64. The information between the generative and performance analysis pipelines (Honeybee plugin) are exchanged through layers. For each type of space or building, a corresponding layer-name is generated and the geometric elements are assigned to that layer. The analytical script uses the layer-name to retrieve geometric entities and run the analysis.

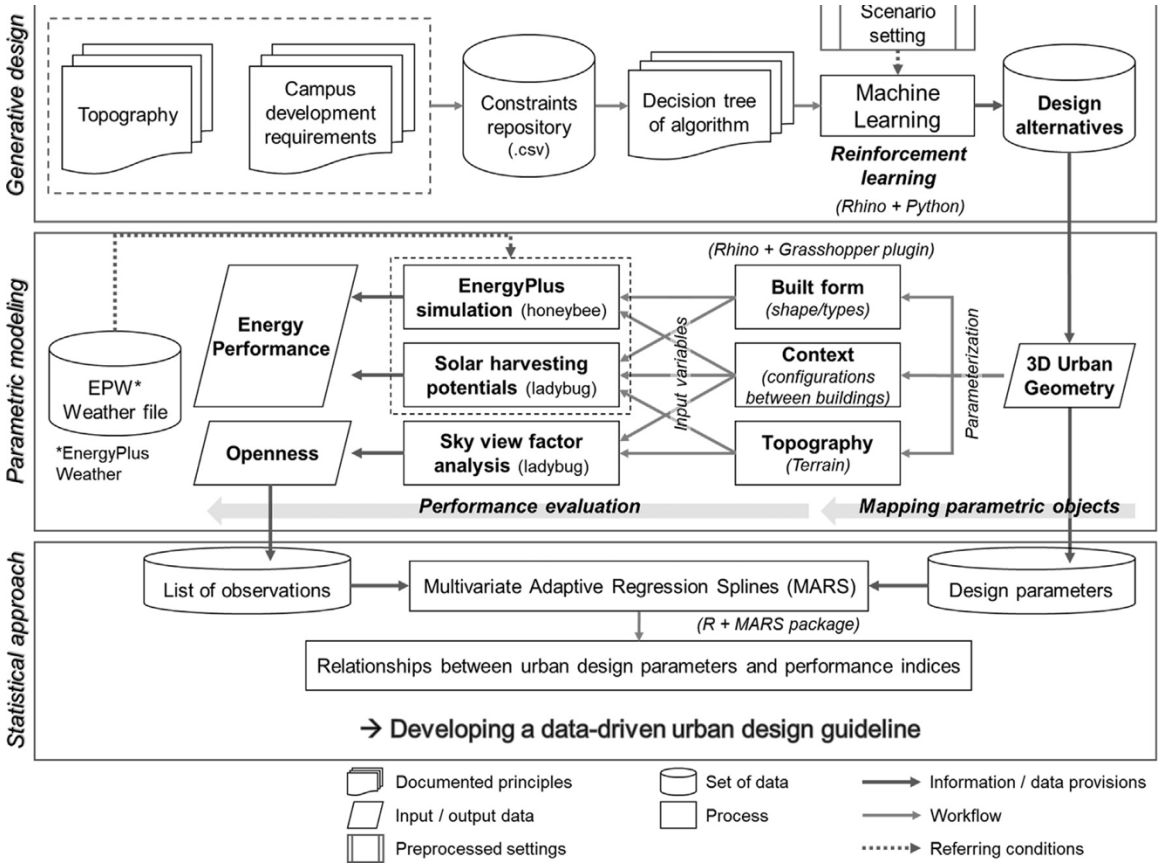


Figure 64. Connecting the generative and analytical components

6.2.1.3 Analytical Results

The analytical modules are applied to the generated scheme and used to populate a spreadsheet (CSV file). The analyst was able to utilize the generative solution and without requiring additional expertise. Since the buildings housed specific functions based on program requirements (Table 44), a default building schedule, lighting, and equipment loads for the floors are used for analysis. The analysis was conducted on three types of schemes (a) concentrated (b) clustered and (c) decentralized. The concentrated formations

are based on a predetermined urban design layout where the generative algorithm is used to ensure the fulfillment of area requirements. The clustered formations resulted from restrictions imposed on buildable regions with specific changes in slope values. This led to a clustering of building formations in specific locations of the site. The decentralized design schemes are a result of relaxing the slope constraints so that the building massing can be spread over the site. The data collected from these studies are shown in Table 45.

Table 46. Multiple performance analysis results and subsets of design parameters for 30 campus design alternatives.

Scenarios	Sky View Factor (%)	Average solar radiation (kWh/m2)	Potential solar power (kWh/m2)	Annual energy demands (kWh/m2)	Number of buildings	Number of thermal zones	Coverage ratio	External wall area	FAR
a.1	58.74	409.81	31.39	195.39	51	922	0.17	14057	1.13
a.2	58.74	409.45	31.36	207.52	51	949	0.17	143951	1.15
a.3	58.97	408.72	31.31	217.5	51	896	0.17	14519	1.16
a.4	59.04	409.23	31.35	215.94	51	873	0.17	141961	1.14
a.5	58.67	409.48	31.37	223.17	51	889	0.17	143530	1.14
a.6	58.73	410.39	31.44	220.37	51	989	0.17	143688	1.15
a.7	58.86	415.24	31.81	221.23	51	819	0.17	134800	1.1
a.8	58.64	407	31.18	218.11	51	916	0.17	147379	1.17
a.9	58.89	412.01	31.56	220.94	51	854	0.17	140120	1.13
a.10	58.68	409.35	31.36	225.34	51	908	0.17	147060	1.17
b.1	54.17	591.65	45.32	167.86	48	274	0.12	83919	0.68
b.2	52.71	565.31	43.3	165.17	54	287	0.13	89440	0.71
b.3	54.32	580.68	44.48	171.8	48	345	0.12	81440	0.85
b.4	52.96	566.79	43.42	174.24	57	404	0.14	96160	1
b.5	53.9	574.36	44	171.03	54	369	0.13	87600	0.91
b.6	55.18	392.72	30.08	164.97	65	672	0.16	158960	1.66
b.7	53.48	408.17	31.27	170.82	69	701	0.17	166960	1.73
b.8	53.82	405.32	31.05	171.49	71	682	0.18	159720	1.68
b.9	52.3	404.98	31.02	167.9	83	745	0.2	176160	1.84
b.10	53.48	415.05	31.79	166.01	66	687	0.15	163360	1.7
c.1	46.36	552.32	42.31	177.95	74	699	0.18	162320	1.73
c.2	44.54	543.87	41.66	176.68	82	746	0.2	176640	1.84
c.3	44.17	569.32	43.61	179.37	74	692	0.18	166400	1.74

(Table 46 continued)

c.4	42.49	521.34	39.93	169.09	89	533	0.15	117500	1.32
c.5	48.07	571.6	43.78	175.58	63	626	0.15	147680	1.55
c.6	46.1	583.69	44.71	168.6	64	648	0.16	154560	1.6
c.7	47.85	601.62	46.08	176.23	66	542	0.16	127280	1.34
c.8	45.27	593.17	45.44	175.45	72	738	0.17	151840	1.58
c.9	47.02	594.23	45.52	171.75	61	608	0.16	144640	1.5
c.10	45.32	579.28	44.37	179.69	66	726	0.16	173360	1.79

An analysis of the performance of the three types of studies revealed that clustered layouts consume the least annual energy per unit area, while the concentrated plans require the least energy to function and the decentralized class of layouts possesses the highest potential for solar power generation. Figure 65 shows the average performance and provides a comparison between the scenarios or classification of the schemes.

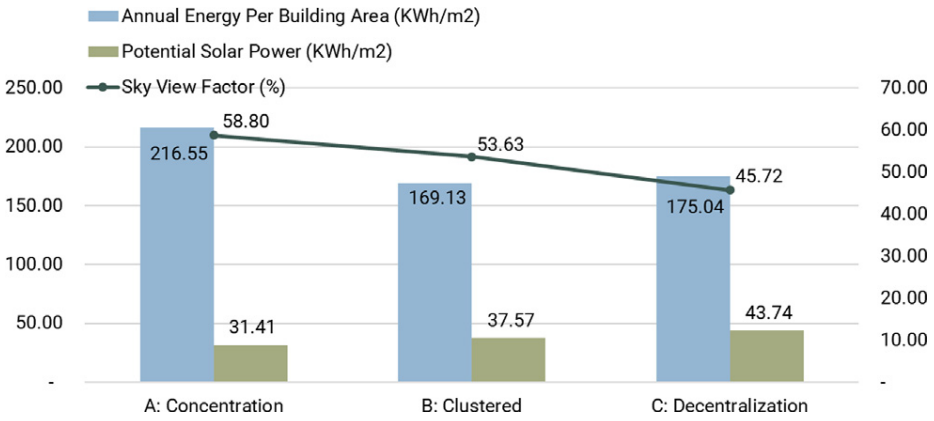


Figure 65. Average performance for 10 alternatives on each scenario.

6.2.1.4 Correlation between design parameters and performance

The correlation between design parameters and performance analysis is developed from the analytical components to provide guidelines or recommendations for the subsequent design development. In this study, the analyst developed Multivariate Adaptive Regression Splines (MARS) to analyze the nonlinearity among variables and predictors by forming a piecewise linear model in R (software). Table 46 shows the variables used in the regression models of each predictor and the generalized R-squared values which is the estimate of the accuracy of the prediction. Four estimates were provided as a result of this analysis. They are as follows:

i. The sky-view factor was estimated by the number of buildings, the number of thermal zones, building coverage ratio, external wall area, and floor area ratios. (Figure 66)
ii. The potential for generating solar power is estimated from the number of buildings, external wall area, and sky view factor. (Figure 67)
iii. The annual energy demand is estimated using the external wall area, and sky-view factors. (Figure 68)
iv. Energy balance is computed as the potential solar power divided by the energy demands and estimated using the number of buildings, thermal zones, and sky-view factor. (Figure 69)

Table 47. Model variables for each predictor and R-squared values

Response	Predictor Variables						
	Number of Buildings	Number of thermal zones	Coverage Ratio	External wall area	Floor area ratio	Sky-view Factor	R^2 Value
Sky-view factor	✓	✓	✓	✓	✓	X	0.894
Potential solar power	✓	X	X	✓	X	✓	0.952
Energy Demands Energy Balance	X	X	X	✓	X	✓	0.94
Energy Balance	✓	✓	X	X	X	✓	0.956

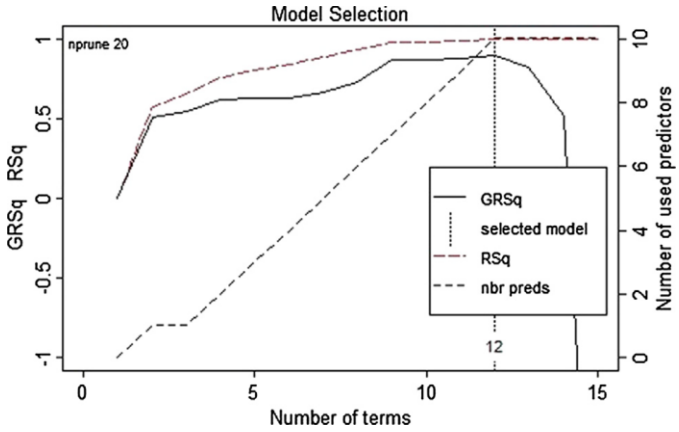


Figure 66. Selection plot for sky view factors prediction model

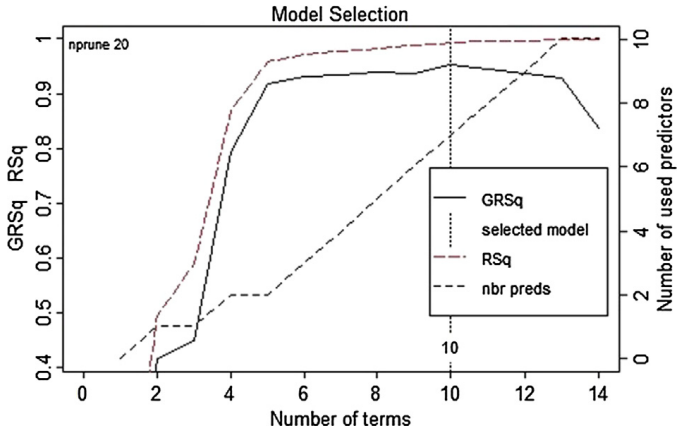


Figure 67. Selection plot for the solar potential prediction model

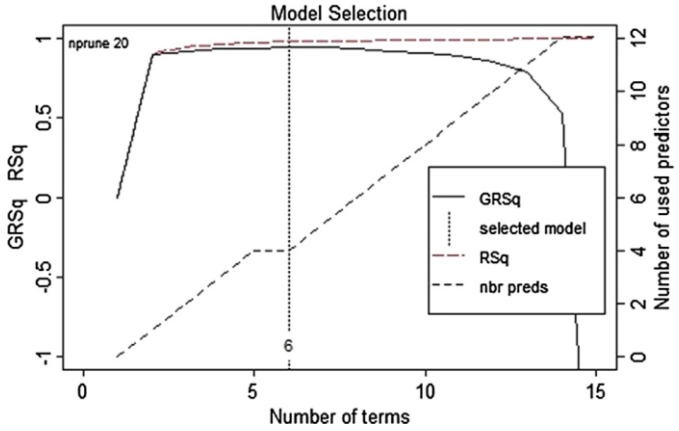


Figure 68. The selection plot for energy demands prediction model

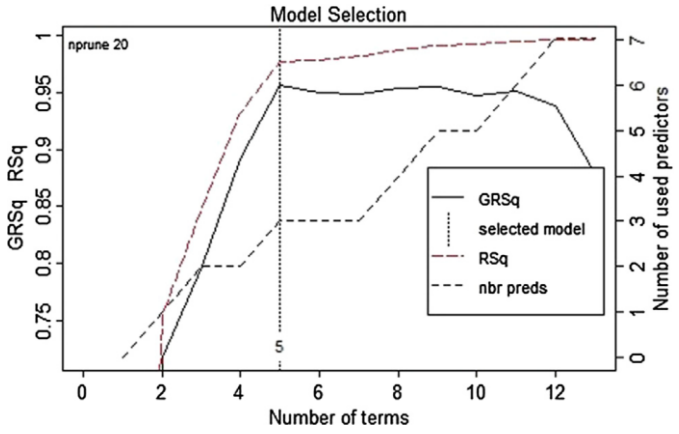


Figure 69. Selection plot for energy balance prediction model.

6.2.1.5 Conclusion of this study

This study provides a new approach to campus design by simultaneously generating numerous design options and performance analysis using a computational pipeline with minimum human intervention. The findings from the MARS analysis are as follows:

i. When the number of buildings is more than 54, sky view factors decrease. The potential for solar power and energy balance decreases when the number of buildings is greater than 64. Based on this analysis, the inference is that the number of buildings should be between 54 and 64.
ii. When the thermal zones decrease below 687, the sky-view factor increases, and the energy balance decreases. On the other hand, if the number of thermal zones increases beyond 687, the sky-view factor decreases, and energy balance increases. The reduction in the sky-view factor is recommended to reduce the discomfort due to heat and improved energy balance. Consequently, 687 is the recommended number of thermal zones.
iii. The building coverage ratio is recommended to be a value greater than 0.17 for optimal solar potential.
iv. The recommended threshold for the sky-view factor is 54.17% to maintain the optimal energy balance.

6.2.2 Generate-Evaluate Process in Architecture

Rezzae et. al, (2019) provides a detailed account of the research. A summary has been provided in the following sections to demonstrate the utility of generative techniques in performance-driven research and practice of architecture.

The generative problem for a K-12 school prototype was addressed by a customized computational model based on a template provided by the designers. This is a form of deliberate organization of spaces along a circulation spine with additional project-specific constraints described in section 3.2.2. It is used to support the research in decision-making frameworks and process-oriented research that utilizes computational methods to explore constraint-driven design alternatives.

6.2.2.1 Problem setup

In this case, the computational model was developed based on program requirements (Table 47 and 48). Additional constraints of window-wall ratio and orientation were included (Table 46) for analysis. Project-specific constraints based on discussions with designers (Figure 70) were encoded into a decision tree (Table 49). This decision tree was used to generate exhaustive configurations (Figure 71) of a three-dimensional layout of the school.

Table 48. List of spaces and the constraints for the Savannah K-12 case study.

Program	Max Area SF	Min East-West Length	Max East-West Length	Min North-South Length	Max North-South Length	Min Stories	Max Stories
Admin and Media Center, Clinic, Elevator	10,000	40'-0"	250'-0"	40'-0"	250'-0"	1	2
Art, Art Kiln, Art StorageSkills, Business Lab, Tech Lab, Resource, Elevator, Toilets, Storage)	10,000	40'-0"	250'-0"	40'-0"	230'-0"	1	2
Classroom Block (10 Classrooms, Restrooms, Stair Core)	13,000	61'-6"	85'-0"	150'-0	230'-0"	1	2
Classroom Block (10 Classrooms, Restrooms, Stair Core)	13,000	61'-6"	85'-0"	150'-0	230'-0"	1	2
Classroom Block (10 Classrooms, Restrooms, Stair Core)	13,000	61'-6"	85'-0"	150'-0	230'-0"	1	2
Classroom Block (10 Classrooms, Restrooms, Stair Core)	13,000	61'-6"	85'-0"	150'-0	230'-0"		
Classroom Block (10 Classrooms, Restrooms, Stair Core)	13,000	61'-6"	85'-0"	150'-0	230'-0"		
Classroom Block (10 Classrooms, Restrooms, Stair Core)	13,000	61'-6"	85'-0"	150'-0	230'-0"		
Circulation L1 (Corridors, Entrance Lobby) & L2	12,000	NA	NA	NA	NA	1	2
Gymnasium	10,000	80'-0"	80'-0"	125'-0"	125'-0"	1	1
Band Room (Toilets, Storage, Offices, Comm, Elec)	5,360	55'-0"	97'-0"	55'-0"	97'-0"	1	1
Vocal (Office, Storage, Mech, Toilets)							
Dining (Student Dining, Teacher Dinning, Lockers, Storage, Toilets, Kitchen, Serving Area, Custodial, Mech, Elec, Office, Stage Platform and Chair Storage)	13,630	94'-0"	145'-0"	94'-0"	145'-0"	1	1
Total Area Target SF	139,000						

Table 49. Savannah K-12 investigated variables.

BFG Massing	WWR	Orientation	Total Options
54	4	5	1080
1 to 54	25, 30, 35, 40	0, 45, 90, 135, 180	

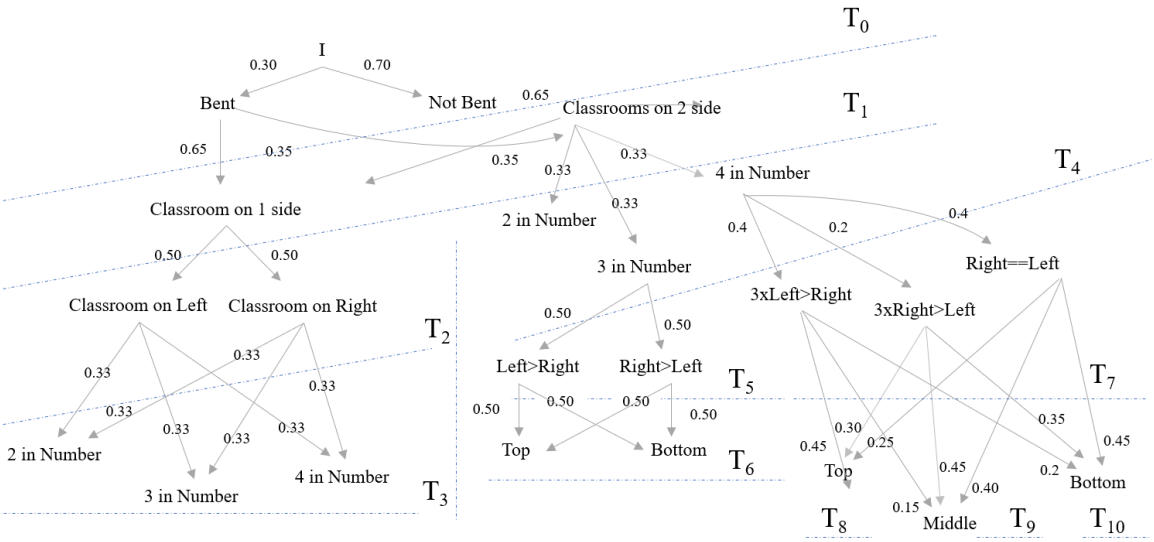


Figure 70. Discussion with designers (Perkins and Will research group)- sample of classroom block organization.

Table 50. Constructing the Decision Tree -sample of classroom block organization.

	Decisions	Name	Expression	Element of Tree
0	Bent or not	T0	T0 = (x≤0.3 ∨ x>0.3)	(∅)
1	Classroom on 1 or 2 side	T1	T1 = (x≤0.5 ∨ x>0.3)	(∅)
2	On Left or right	T2	T2 = (x≤0.5 ∨ x>0.5)	(T1)
3	2,3 or 4 blocks	T3	T3 = (x≤0.33) ∨ (x>0.33 ∧ x ≤ 0.66) ∨ (x>0.66 ∧ x ≤ 1.00)	(T1)
4	2,3 or 4 blocks	T4	T4 = (x≤0.33) ∨ (x>0.33 ∧ x ≤ 0.66) ∨ (x>0.66 ∧ x ≤ 1.00)	(T1)
5	Left > right	T5	T5 = (x≤0.50 ∨ x>0.50)	(T1∧ T4)
6	Top or bottom	T6	T6 = (x≤0.50 ∨ x>0.50)	(T1∧ T4)
7	3 on right, 3 on left or left==right	T7	T7 = (x≤0.40) ∨ (x>0. 40 ∧ x ≤ 0.60) ∨ (x>0.60 ∧ x ≤ 1.00)	(T1∧ T4)

(Table 50 continued)

8	Top	T8	T8 = (x≤0.45) ∨ (x>0. 45 ∧ x ≤ 0.75) ∨ (x>0.75 ∧ x ≤ 1.00)	(T1∧ T4 ∧ T7)
9	Middle	T9	T9 = (x≤0.15) ∨ (x>0. 15 ∧ x ≤ 0.60) ∨ (x>0.60 ∧ x ≤ 1.00)	(T1∧ T4 ∧ T7)
10	Bottom	T10	T10 = (x≤0.20) ∨ (x>0. 20 ∧ x ≤ 0.55) ∨ (x>0.55 ∧ x ≤ 1.00)	(T1∧ T4 ∧ T7)

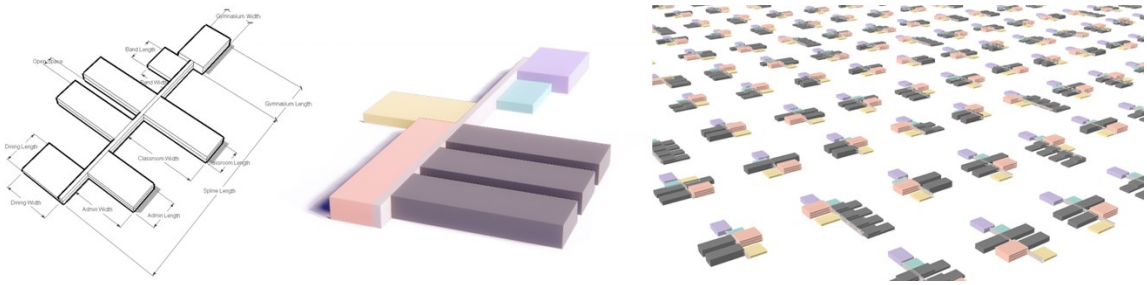


Figure 71. (a, b) Parameterizing the building massing for school, (c) Building forms randomly generated for Savannah K-12 School using a customized algorithm

6.2.2.2 Methodology

The analytical research team used the Design Space Construction (DSC) (Haymaker et. al., 2018) methodology to utilize the computational (parametric) model and analyze the performance of various alternatives. Then the interface layer is used to extract the input data and collect the results. Finally, the data visualization layer gathers stakeholder input preferences and plots the data of every alternative in the design space. The study was applied to generate guidelines for architects regarding the daylighting and energy efficiency of the building.

The objective of the analytical study is to achieve higher daylighting while minimizing energy consumption. Energy Use Intensity (EUI) is used as an energy performance indicator. Energy Plus was used for the simulation. Radiance is used to simulate daylighting with an illuminance level between 300 to 2000 lux, twice a year for regular and irregular occupancy. Additional assumptions (Table 50) are required to allow the analysis and recommendation process because actual patterns of occupancy, usage, materiality, specifications of mechanical systems, etc., are not known at the early stages of the design process.

Table 51. Assumptions for Savannah K-12 case study.

Variable	SI units	Assumption
Roof R-Value SI		3.671
Wall R-Value SI		1.421
Window U-Value SI		3.972
Window SHGC		0.25
Window VT		0.5
Floor R-Value		3.389
Exposed Floor R-Value		3.389
Skylight R-Value		0.271
Air Changes Per Hour	ACH	0.6
Ventilation Rate per Area	m^3/ m^2s	0.0006
Ventilation Rate/Person	m^3/ m^2s	0.005
Number of People/Area		0.249
Lighting Power Density	W/ m^2	9.365
Occupancy Schedule		daily
Equipment Loads/Area	W/ m^2	10.97
HVAC heating set point	°C	18
HVAC cooling set point	°C	26
HVAC heating setback	°C	12
HVAC cooling setback	°C	32
Baseline HVAC System		hybrid system

6.2.2.3 Analysis of Generated Layouts

Design variables (Tables 47, 48, and 50) led to the generation of 1080 design alternatives. The design team chose 54 design alternatives selected from the generative process, and the analysts introduced the window-wall ratio, and orientation was introduced by the analysts to generate a set of alternatives for analysis. Figure 72 shows the entire design space as represented by a parallel-coordinates-plot (PCP). The PCP interval values can be manually adjusted to eliminate design options that do not lie within the range specified by the user. It is a highly efficient approach to decision-making because it is interactive and permits the visualization of conflicting results. Figure 73 shows the filtering of alternatives with lower Energy Use Intensity (EUI) and higher daylight illuminance and the correlations between these indicators and higher WWR among the generated solutions. The range of permitted values is adjusted (Figure 74) to reveal the highest value function with EUI of equal or less than 90 kBtu/SF and the average illuminance of greater than 40%. The internal correlation between design parameters of the system leads to optimal configuration with 3 blocks of 2 stories of classrooms. However, other options may achieve individual objectives. It can be inferred that orientation does not affect the performance because all values of orientation exist within the solution range.

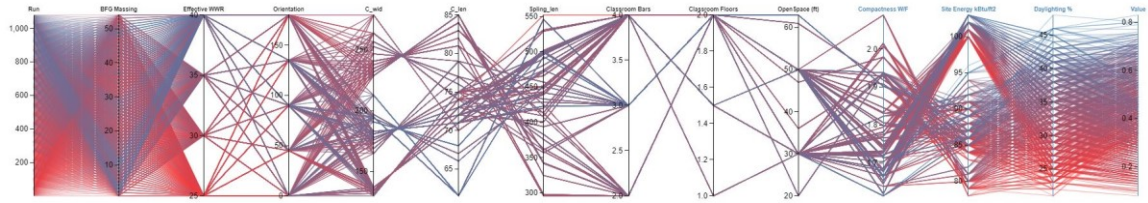


Figure 72. The plot of the design space of 1080 alternatives.

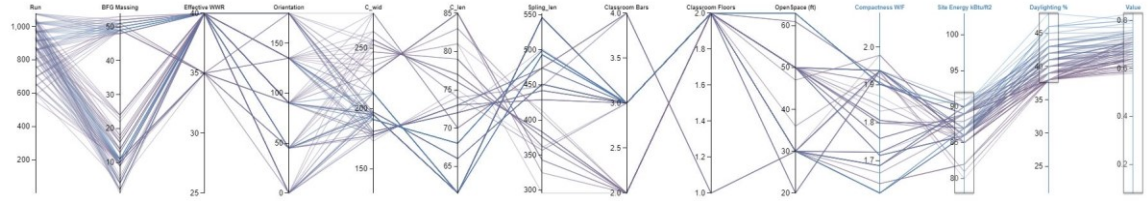


Figure 73. Filtering alternatives within a threshold value looking for correlations.

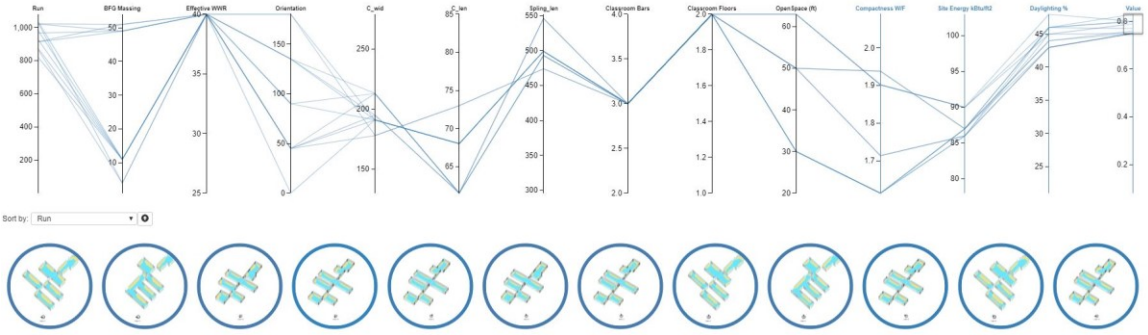


Figure 74. Preferred region of the design space based on the objectives.

The optimal solutions for guiding designers were determined by limiting the design space and analyzing a diverse set of design alternatives. The objective is to provide a viable range of values for design development to ensure appropriate daylighting and energy performance. The results of the study are provided in Table 51.

The recommendations for classroom blocks are supported by the correlation between design parameters of WWR, orientation, and number. This is illustrated by scatterplots provided in Figure 75.

The study demonstrate that the early design phase can be supported by allied design generation and performance analysis that helps in determining the conflicts and bottlenecks in multi-objective decision-making. The study also demonstrates that guidelines allow designers to explore environmentally appropriate solutions without compromising an archetypal solution or concept.

Table 52. Results and recommendations for building configuration.

Design Parameter	Recommended Value/Range
# of Classroom Floors	2
# of Classroom Bars	3
Compactness	Less than 2
Orientation	No significant effect
WWR	40%
Classroom Bar Width	178-213 feet
Classroom Bar Length	61-73 feet
Open Space btw Classrooms	30-63 feet
Corridor Spine Length	474-546 feet

Figure 75 a. Correlation between design variables and window wall ratio (WWR)

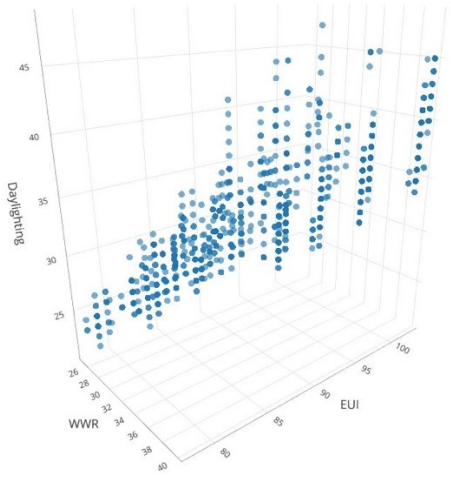


Figure 75 b. Correlation between design variables and compactness

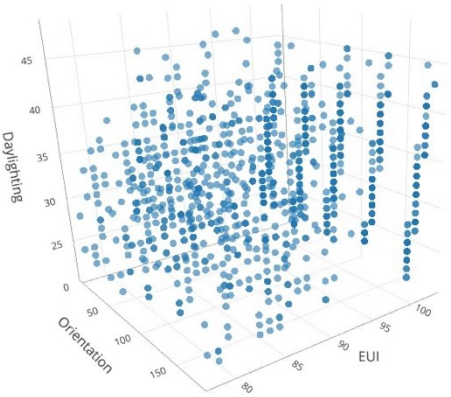


Figure 75 c. Correlation between design variables and number of classroom bars

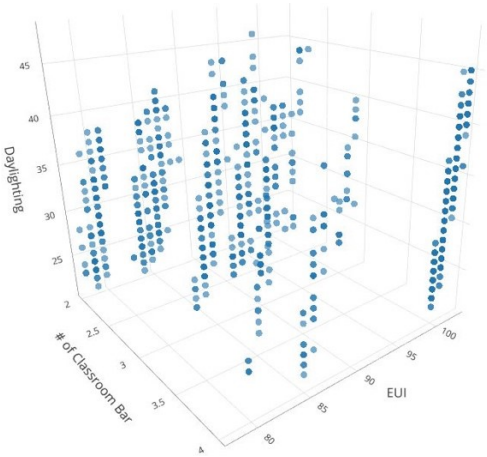


Figure 75. Examples of 3D scatter plots showing the correlations between design variables and two objectives of EUI and Daylighting: (a) WWR, (b) compactness, and (c) number of classroom bars.

6.3 SAP-Problems in the Context of Commonly Used Techniques

Two dominant computational design approaches are used in design practices and research, namely, parametric modeling (Dong, 2007; Schnabel, 2014) and rule-based or graph-grammar systems (Peng et. al. 2016; Muller et. al., 2008). Parametric models address customization of a design scheme whereas rule-based or graph grammar systems are approaches that provide general generative solutions. While these paradigms are eminently useful, they do not address specific SAP-related problems, due to the computational complexity of the problem. Similarly, heuristics (Jagieleski and Gero, 2006; Calixto and Celani, 2015) have been frequently used in SAP-related research but they limit the type of solutions and scope of problems. The following sections discuss the enhancements provided by IDF/PLUGS which are not currently addressed.

6.3.1 Comparison with Rule-Based Systems

Rule-based systems in architecture and urban design are addressed by a three-step proposal (a) collect exhaustive catalogs of patterns of the built-form and (b) develop myriad rules for concatenation of the elements and (c) synthesize new design configurations by applying selected rules on geometric forms. Rule-based systems are described as *formal systems* where the rules were accepted as axioms that could be used to synthesize designs. (Alexander 1964, 1977). Their advancement as procedural graph-grammar systems is used in research and commercial applications such as City Engine (Muller and Parish, 2001).

Despite their merits and contribution to the development of design thinking, rule-based systems cannot be considered axiomatic in architecture and urban design based on the Popperian model (March L., 1976; Keller, 2006) because the necessity for a particular set of shapes or their composition, cannot be justified or verified by data. SAP-related constraints are considered a better estimate of the design requirements because they are typical requirements of a program for which a design problem is solved.

In this thesis, it is proposed that "rules" are symptoms of underlying mechanisms expressed as *context* by Alexander. The reason for a rule to exist is based on constraints of the environment and requirements of the design problem. The rule-based systems are used in the development of IDF for specific typologies of buildings and geometric formations. The rules are embedded in the IDF components and manipulated by the optimization algorithms to meet the specifications required in the program. While rules are significant in developing design solutions, the following scope for improvement has been addressed by the proposed computational framework:

i. Problems formulation: Rule-based systems apply to *any* design problem when an overall scheme is known and possible conditional operators are provided by the user. By formulating the problem based on constraints and geometric operations, IDF/PLUGS are applied to SAP-related problems in architecture and urban design and generate spatial solutions with minimum human intervention. The IDF is generally applicable with fewer pre-processing steps whereas rule-based systems require elaborate preparatory steps before they can be applied.

ii. Techniques: Various optimization techniques and computational geometry algorithms are used to develop IDF whereas rule-based or graph-grammar systems are simplistic because they rely on conditional operators or procedural application of rules. Essentially, IDF *generates* a rule-based system and operate on the rules using the optimization modules. The generation of rules is the development of objectives or constraints from the inputs provided by the user. In a rule-based system, these constraints have to be manually developed.

iii. Applications: IDF generates spatial solutions to a wide range of design problems in architecture and urban design. Rule-based systems and graph grammar produce shapes and compositions that resemble design patterns rather than address design solutions that can be evaluated numerically.

iv. Generalization of geometry: Rule-based systems are limited because spatial solutions to design problems are not limited to a specific set of geometric forms and generalization based on exhaustive catalogs of primitive shapes and rules is not practical. Even if all rules are developed, parsing them, and extracting appropriate rules to apply to a problem is not computationally feasible. IDF provides components that remain unaffected by variations in topology and scale of the geometric formations.

v. Constrained-solutions in design: Rule-based solutions face "combinatorial explosion" which led to the identification of the SAP problems in architecture and urban design. SAP-models address constraints routinely used by designers to develop and evaluate layouts or configurations of buildings (Liggett, 2000;

Homayouni, 2007). SAP constraints and SAT proposed in this thesis are fundamental to the problems addressed and the solutions generated by IDF/PLUGS that provides spatial solutions by optimizing the attributes of the spaces-activity relations in a layout.

vi. Use in the design process: In design problems, due to the topological variations or variations in geometry, it is difficult to re-use or develop general solutions using conditional operators and specific solutions provided by a rule-based or graph grammar formulation. SAP-related solvers such as IDF/PLUGS provide solutions to address a vast range of similar design problems where the SAT is bundled in a form that precludes the need for development and evaluation by the end-user.

vii. Recommendation: Discounting the SAP, rule-based and graph-grammar systems are preferable for exploration of the composition of shapes where numerical constraints are irrelevant whereas IDF type of solvers are constrained generative tools based on advanced optimization techniques that are better equipped to address design problems where the organization is based on specific constraints to address well-known design problems rather than a composition of shapes and forms.

6.3.2 Comparison with Parametric Models

Parametric modeling is applied in design as a boolean synthesis or the state-action paradigm (Mitchell, 1988) that sequentially processes the geometric operations based on inter-connected variables and conditional logic. IDF or PLUGS can be categorized as *generative* tools because unlike parametric models, generative models do not require an

initial scheme from the designer. Rather, a computational framework such as IDF or PLUGS, classified as generative models, operate on constraints and generate spatial output that can be expected from the design process.

The goal of this research is to support the process of design, where the proposed framework, IDF, provides a set of components (section 4.5) that can be connected in various permutations to emulate the spatial output of a vast number of design processes. These components will generate the entire model being studied. Consequently, it will provide greater value to the analytical studies. It will eliminate the need to develop ad hoc models for SAP-related design processes. A comparison is provided as follows:

i. Solution: Parametric models use an initial scheme, provided by the designer, to develop a computational model with varying parts. Whereas IDF is a generative computational framework that provides an optimal solution to a SAP that in numerous design processes based on inputs such as program requirements and bylaws.

ii. Problems addressed: Parametric modeling, as a technique, addresses a vast range of applications from form-finding in buildings to furniture design. Parametric models are frequently used in practice and analytical research to prototype subsets of buildings or urban formations and run simulations in analytical studies to measure building performance or urban characteristics (Anton and Tanase, 2016). IDF/PLUGS are SAP-models that address specific problems in architecture and urban design related to spatial output such as the generation of layouts.

iii. Difficulty-level: The solution to SAP is non-trivial and it is considered an intractable problem because the algorithms and geometric solvers require expertise whereas parametric models are commonly developed by designers using visual scripting tools such as Rhino3d-Grasshopper or Dynamo. For this reason, IDF was developed as a plugin for the Rhino3d-Grasshopper environment such that it can be easily accessed by designers.

iv. Generalization: IDF provides general solutions with several SAT bundled into the models for plug-and-play type usage. IDF operate across scales and topologically variant problems. Parametric models are developed for a design problem and typically, cannot be reused unless the designer uses the same solution or design motif because the relations between geometric forms, type of input, and output is hard-coded.

v. Design Exploration: IDF provides a systematic approach to conduct design exploration whereas parametric models provide stochastic variations. Since parametric models are based on an initial scheme and use stochastic variables, the scope for generating variations is limited.

vi. Use in Design Process: IDF is designed to support design processes such as floor plans, site plans, and interactive space-activity relations in city-blocks. Since the SAT used in IDF components are unaffected to variations in topology and scale, they have a greater potential for application in known problems of design processes. Parametric models show a greater potential to prototype ad hoc design ideas such

as forms and shapes of buildings. In terms of problems addressed, the parametric models have a greater scope because.

vii. Recommendations: IDF/PLUGS type of solvers are useful in the early stages of design processes to experiment with *overall* solutions of standard problems in design processes and develop schemes to evaluate inputs and constraints. Parametric models are useful in developing detailed solutions to design problems when the designer has a pre-determined scheme to evaluate stochastic variations.

6.3.3 Optimization: Heuristics and reinforcement learning

Since design problems are ill-defined, heuristics are used in space planning and generative design. They include evolutionary algorithms, genetic algorithms, genetic programming (Rosenman and Gero, 2009; Michalek, and Palambros 2002; Rodrigues et al, 2013). A crucial factor in organizing space-activity relations is the flexibility of constraints. Since design decisions evolve, topological or geometric constraints require updates during the design process. To address the continuously changing constraints, an agent-environment interaction process is adopted such that the algorithms can halt and update without perturbing favorable substructures. Apart from the reinforcement learning algorithms as the primary solver, heuristics are used when applying for instance locating the footprints in the site feasibility study. A novel feature is developed to facilitate the interaction between optimization modules, input interfaces, and user interaction. Using state-action and policy updates, the following features were addressed by this research:

i. *Continuously Changing Constraints (Environment)*: This allows users to constantly update inputs and generate solutions by sub-sampling the structure of the solution such that partial configurations are updated without disturbing optimal substructures. This is facilitated by a state-action reward or by updating a policy when the topological structure is used.

ii. *Dynamic Process*: The proposed techniques track the relationship between individual actions performed by the agent and its effect on the layout (Figure 1, p 35). They exploit sequential structures of data which results in the preservation of information about favorable configurations (Sutton and Barto, 1998). The underlying mechanisms result in the search for an overall solution by studying the effect of individual behavior over the entire configuration. Sequential decision making, drawing inferences, and propagating optimal sub-structures have been used in this research to develop the optimization mechanisms.

iii. *Scaling*: To address the scaling issues, function approximators like decision trees and neural networks are used. Using the topological model and numerical reward-formulation, a large number of state-action rewards driven by McMC methods are calculated. This is used to predict new state-action-rewards and minimize an error function – train the model/learn which leads to efficient calculations for future actions. Although the dynamic programming approach is intuitive and used to explain the Q-learning algorithm in section 3.3, it is limited by scale.

6.3.4 Choosing an Approach

This research specifically addresses the SAP and provides solutions, that are enhanced in many ways (Chapter 5). The rule-based systems and parametric models are generic problem-formulation techniques that may be applied to design problems that do not involve the SAP. Based on the experiments conducted in generative systems, rule-based systems prove to be useful when a scheme is presented and rules can be extracted to generate the variations. Parametric studies are useful in developing temporary partial models where random variations of an entity is required. Rule-based systems and parametric models have hard-coded logic which makes it difficult to generalize over non-identical SAP problems. In comparison, the solutions provided by IDF components contain logical arguments which generalize the solutions over various types of constraints and geometric inputs. A quick comparison between the three methods are proivded in Table 53.

Table 53 – IDF specifically addresses SAP

Legend True ✓ False X Partial o	Rule-based	Parametric	IDF
Numerical & Geometric Relations	✓	✓	✓
For SAP	x	x	✓
Generic Models (starting conditions)	x (generate *all* rules)	x (develop code)	✓
Standardized Constraints	x	x	✓
Optimization	x	o	✓
Applications	x	✓	✓
Generalization	x	x	✓
Variations	✓	✓	✓
Project-specific Customization	x	✓	✓

CHAPTER 7. CONCLUSION

This research provides a prototype for an open-ended interactive computational framework that generates spatial output to address SAP-related design problems based on factors such as the geometric and topological variance in design problems, scales and objectives, the difference in context, needs of the user, and aspirations of the designer. The summary of the development of IDF, a prototype, is provided in section 7.1. The contributions and significance of the proposed solutions to SAP are noted in sections 7.2 and 7.3. The development of IDF implies that an alternative design framework may be layered on top of industry-standard CAD/BIM software to enhance allied research in building simulation and data-driven design processes. Such a layer allows interactive structured design processes, which, stores and retrieves information or decisions that lead to the spatial output (section 7.4). Finally, the computational framework is open-ended because the organization of spaces in the domain of architectural and urban design is not exhaustive and the scope for further development is discussed in section 7.5.

7.1 Summary of Objectives and Methods Used

In the prior sections, the thesis proposes that SAP are a specific type of problem in architecture and urban design. The problem along with technuiques to solve them (Table 54) are demonstrated using a computational prototype. The illustration of results highlights the features (Table 2, section 1.4) necessary to apply such solutions in research and

practice. These features are used to assess the scope for improvement in computational techniques that were proposed in prior research (section 2.9). The hypotheses regarding problem-formulation and techniques to address SAP are implemented by mapping computable tasks to methods such as encoding predefined patterns, the optimization of geometric operations, constraint-satisfaction methods in the topological representation of layouts, etc. These methods are collectively described as Space Allocation Techniques or SAT and bundled into numerous components and integrated into a computational framework, namely, the Integrated Design Framework or IDF (sections 4.6). The proposed models operate on standardized constraints that control the solutions (Table-14, p 120). An intuitive input interface is integrated into the modules to accept standardized inputs from the user and mechanisms for user interaction with the SAT were facilitated to develop interactive workflows with the designer. The flow of information from the workflows to model-free optimization techniques (sections 3.1 and 3.2) allows users to update constraints at runtime and dynamically evolve a design solution (case study A, section 5.4, 5.5).

The prototype of the proposed computational framework, namely IDF, demonstrates the application of models to achieve the objectives. Numerous case studies are presented and test cases are developed to illustrate the potential of computable models in research and practice. IDF generates spatial output for SAP-related problems across the three scales, namely floor plans, site plans, and large-scale interactive block planning, which is the primary objective of this research (section 1.2).

Table 54 describes the proposed features of SAP and corresponding techniques that were experimented or incorporated into the framework of IDF and PLUGS. While this is merely indicative of the possibilities, the research demonstrates the structure and range of solutions that can be accommodated by generative design formulation. The solutions proposed in this thesis allow an artificial system to generate solutions by processing constraints and using pre-determined operations to meet the constraints.

Table 54. Features and elements of the Space allocation and techniques or factors considered to achieve the attributes. Auxilliary objectives introduced in Table 2

Features of Space Allocation			Elements considered	Used
1	1.a	Geometric Constraints	Geometric attributes of spaces/location Matrix of Area Proportions of length to width Dimensions of length, width Orientation	✓ ✓ ✓ ✓ ✓
	1.b	Topological Constraints	Match fixed space to a fixed location Adjacency between spaces Proximity to Perimeter Circulation/connectivity between spaces	✓ ✓ ✓ ✓ ✓
	1.c	Procedural Constraints	Rules on Geometric Shape / Form Byelaws & Prescriptive guidelines Circulation (hierarchy) Graph Algorithms: centrality, dispersal, etc	✓ ✓ ✓ ✓ ✓

(Table 54 continued)

2	2.a	Methods for Geometry Generation	Varying Dimensions Grouping cells in a grid Tessellation Medial axis transform Subdivision of boundary Graph Grammar Procedural	✓ ✓ ✓ ✓ ✓ ✓
	2.b	Generalization of shapes	Rectangles, orthogonal polygons Predetermined rules for valid shapes General polygons and curves	✓ ✓ ✓
3	Methods for Optimization		Decision Tree (Search) Mathematical programming: LP, MILP Iterative: hill-climbing Heuristics: evolutionary algorithms, GA, GP Markov chain Monte Carlo methods Reinforcement Learning	✓ ✓ ✓ ✓ ✓ ✓
4	Input Interface		Dynamic constraints: equations & logic Spreadsheet formats like .csv, .xlsx Access to numerical and boolean values Interprets inputs to generate constraints	✓ ✓ ✓ ✓
5	User-interaction		Update solutions at runtime Propagate updates to the input Reset system to update	✓ ✓ ✓
6	Application/scale		Floor plans Parcels & Massing, Feasibility Studies Spatial Networks	✓ ✓ ✓
7	Design Automation Framework		Association between modules to support a comprehensive design process. Modules for the individual use-case. Extensibility of framework.	✓ ✓ ✓

(The Table 54 describes the constraints and methods of the space allocation process. They include prior research and elements that were used in this proposal.
1-4 were informed by prior research to develop the solution for SAP.
1[c], 2[c] and 5-7 are proposed to extend the solution.
5 has been expressed as a desirable feature in prior efforts)

7.2 Contribution

At present, typical computational solutions (CAD/BIM/GIS) in architecture and planning provide documentation, which is essential for construction and facilities management, but the support to design processes can be improved. This research provides computational models for constrained space allocation in fundamental design problems. They generate design solutions from standard constraints used by designers. It may be described as an approach that generates solutions from first principles. Based on case studies and illustrations provided in chapter 5, the main contributions of this research are:

i. A solution to SAP/computability in design processes: The proposed solution to the SAP has been demonstrated. The solution applies to a wide range of geometric forms and topologically variant problems.

ii. Sequential Optimization: Model-free techniques have been successfully introduced to track favorable substructures and the effect of constraints. It is synchronized with the input interface such that the optimization process is paused while the updates to the input change the objective functions. After the update, the optimization process is resumed without loss of information regarding the optimality of sub-structures because the agent scans substructures and notes the effect of displacement of spaces

before updating the partial layouts. This approach allows the sequential processing of each change is an input.

iii. SAP Across Scales: The components of the IDF framework are connected to form workflows that replicate complex design processes, operating across scales and distinct domains of design practices, segregated by expertise and specialization. The IDF workflows allow designers to generate solutions that flow linearly from the organization of space -activity relationships in city-blocks to the floor plan.

iv. Generalization of Design Problem: At each scale, design problems vary in terms of geometric inputs, topology, inputs, or intricacy of the layout. These aspects were addressed by SAT and connectivity between components which allow the user to generate complex spatial output to SAP.

v. Design Exploration: Three types of design exploration, embedded in IDF, allow users to exhaustively search the solution space and systematically review the range of possibilities. Using the proposed models, designers can explore the propagation of constraints for analysis or generate the spatial output of design problems from constraints provided by the user. Alternatively, designers can utilize constraints generated from pre-design processes that rely on data analytics.

vi. User-interaction: IDF provides a standardized user interface with intuitive input-mechanism that converts the user-inputs into constraints that are converted into objective-functions for the optimization process. The input interface eliminates the need for expertise or knowledge of the SAT used in IDF while simultaneously

allowing the user to make changes to objective functions at runtime, which is solved by the algorithms.

vii. Reusable/Modular workflows: The IDF workflow contains the information for a set of design decisions and constraints used by the designer to generate a solution. Since the *IDF* components are algorithms of design processes, the workflow for a design problem can be re-used with project-specific constraints and allow a continuous evolution of the solution rather than conduct similar explorations for each project. A designer can start the design exploration from a previous workflow that is developed for a similar problem and simply update the constraints. It provides a scope to continually improve the workflow by evaluating prior projects and exploring new components. Existing workflows can be altered in various ways (sections 4.4 and 5.9) which allows a constant evolution of the design problem. This implies that fundamental processes, not just shapes, and composition, but rather the design process is replicated and stored in memory.

Table 55. Features of techniques

Compared Features of S.A.P. True ✓ False X Partial o			*Salient Features of Prior Art* A. Varying Dimensions / Mathematical Programming B. Dual of a Graph C. Cell Grid Formulation D. Rule-Based Systems / Graph-Grammar E. Boundary Partition F. Physics Based Formulation G. Urban Forms and Networks							
			A	B	E	C	D	F	G	**IDF**
1	1.a	Geometric Constraints	x	x	✓	✓	x	✓	x	✓
	1.b	Topological Constraints	✓	✓	✓	✓	x	x	x	✓
	1.c	Procedural Constraints	x	x	x	x	o	o	x	✓
2	Generalization of shapes		x	x	x	x	o	x	o	✓
3	Provision for Optimization		✓	✓	✓	✓	x	x	x	✓
4	Input Interface		x	x	x	x	o	o	✓	✓
5	User-interaction		x	x	x	o	o	o	o	✓
6	Application/scale		x	x	x	x	x	x	x	✓
7	Integrated Design Framework		x	x	x	x	x	x	x	✓

7.3 Significance of This Research

The necessity to generate design alternatives for building simulation and analysis were the initial drivers for this research. The research evolved upon a closer examination of the problems and led to wider applications in practice, with the development of a comprehensive design solutions. The following sections indicate the significance of computation in contemporary research and practice.

i. The flexibility of the design process (experimentation & exploration): The challenge to computation in design is that design processes are surprisingly flexible and design problems are not identical. The flexibility in design processes is due to the constant alteration in geometric and topological constraints during the inception phase. While the novelty of form is an unstated objective in design processes, the spatial output is considered if it meets typical constraints (Kelley, 2006). Designers experiment with shapes and forms because bylaws and contemporary construction techniques allow the designers to experiment with a vast range of spatial organization, but not all schemes are optimal. The vast possibility of design alternatives can be effectively explored using IDF workflows as illustrated by the proposed SAT that provides the means to generalize geometric forms and simultaneously constrain the solutions.

ii. Data-Driven design process enhances the scope of allied research: Information (data) from various sources can be used to inform the proposed models. The user may utilize various analytical techniques to set up the constraints which govern the placement of spaces and generate the geometry of the layouts (section 6.1).

Alternatively, IDF workflows, using basic constraints, generates a large number of variations that are used to measure building performance. It is proposed that IDF components can be used to generate the models for simulation, from which guidelines and constraints are developed and used as inputs to computational models to generate the updated solution (section 6.2). This is based on March's vision for computation in design (section 1.5).

iii. Context Development: Design processes in a city operate in silos where buildings are developed independently, following the guidelines of an overarching scheme. Planners and urban designers provide schemes with prescriptive rules, bylaws, and guidelines used to regulate buildings which cannot be sufficiently developed for local optimization of buildings and individual parcels. This research has pursued SAP across three design processes and resulted in the formulation of computational models of design problems that were integrated using shared data structures and methods. In IDF, a topological representation of the SAP is proposed across design processes. The representation is utilized to solve design problems using a central topological representation that is operated upon and subsequently, the process branches out into geometric formulations for various scales and specific configurations. This formulation of IDF components permits intricate workflows that can generate permutations of spatial output across various scales of anticipated development. The application of IDF allows the hypothetical development of various scenarios across three scales from the design of city-blocks to site planning, parcellation-massing, and space planning for floor plates of each building mass.

7.4 Implications – An Alternative Design Environment

The thesis proposes *an alternative design environment* where the potential for semi-automation and structuring the design information is implicit in the workflow. The prototype is an interactive layer of the SAP solution, which is incorporated with CAD/BIM software. The alternative environment integrates the design logic with CAD-algorithms and structures the information of spatial output using the principles of BIM. The computational design processes proposed in this thesis differs fundamentally from established methodologies such as rule-based systems or parametric models because the IDF components address design problems, namely SAP, rather than providing solutions to instances of problems. IDF operates on SAP constraints and provides architectural solutions rather than indicative illustrations of hypothetical schemes or alternatives on a theme. Unlike other paradigms of generative design, which address a specific solution, the proposed solutions do not rely on a pre-determined scheme or relations. In IDF workflows, a general optimization algorithm guides the geometric operations such that constraints are met. This optimization apparatus and geometry generators are independent to each other. Thus, by using the SAP to address design processes, this proposal provides a solution to a commonly recurring design problem, rather than solving an instance of a problem.

Semi-automation of a design process may be described as a computer-controlled environment where the user and the software work simultaneously on a design problem such that the designer's task is connected to actions taken by the underlying algorithms. Semi-automation entails the development of models with embedded relations that are exposed to the user at runtime such that the relations can be updated to generate appropriate

architectural or urban design schemes. It requires modifications to the computational pipeline that connects the user interface with the optimization algorithms and generative processes. The case studies and test-cases illustrate the dynamic interaction between the designer and the software where the user can alter basic inputs on a spreadsheet or governing shapes at runtime and the models update the internal organization of spaces and their activities. Although the solution is a prototype, various case studies and complex test-cases demonstrate semi-automation in design processes.

The components of the IDF prototype target design problems, not just shapes, and rules of composition, consequently, an IDF workflow consists of constraints and operations to generate the spatial output of a design process. The IDF workflow is an electronic document that contains all the information required to generate the spatial output, which can be saved and retrieved with a guarantee that it will generate the same solutions, consistently, with stochastic variations if programmed to do so. The components of the workflow, which contain the generative logic and the sequence of connections between them (operational logic) can be examined, and modified to replicate the solution or alter it suitably. This application of IDF workflows leads to an explicit design process that allows workflows to be manipulated in the same way as any physical system rather than an inscrutable design process (CAD/pencil0-sketches) where the output conceals the details of the process that leads to the design. It is anticipated that the explicit design process will enhance the dissemination of design knowledge across the design teams and lead to the structuring of information about the process.

7.5 Scope for further research

The scope for immediate improvement in the proposed solutions is related to the implementation of the models. It can be enhanced by appending to the set of components and incorporating knowledge-domains close to the optimization algorithms. A sample of the possible enhancements are as follows:

i. Enhancements: Numerous models for massing typologies can be added to enhance the scope of the framework. The alternative geometric organizations have to be programmed to include typologies or geometric patterns.

ii. Urban Design Solutions: Models for the generation of urban form are limited in scope. They can be enhanced to include additional features for analysis and generation from input fields used by researchers. Minimum fields have been used to indicate the possibilities of a generative solution.

iii. Data-Analytics: Statistics related modules that are required in conjunction with the constrained generative processes to analyze the solutions and track the changes in output. At present, data analytics is delegated to the user, but the platform can be developed to integrate the generative solutions with analytics.

While the limitations are technical issues that will evolve is the research is pursued, the scope for further is related to artificial intelligence in design. There are two major directions for future research based on the proposed research:

i. Recommendation: Recommendation features can be developed to guide the user in generating constraints by providing a set of constraints that are most suitable from

prior projects. It is inference-based reasoning systems that require additional research. Apart from inputs, the provision for recommendation systems may include workflows of IDF components that are stored and retrieved. Thus the system can propose tentative solutions to the designer at the inception of the project.

ii. NEW Solutions: A new solution does *not* imply a rule-based concatenation of geometric forms and it is *not* a purely random process of placing shapes. Rather, generating new architectural solutions implies that the artificial design process determines a new set of geometric operations that are not programmed *or* it invents an organization of spaces that are not governed by a pre-determined set of conditional operators.

BIBLIOGRAPHY

Agrawala, M., Phan, D., Heiser, J., Haymaker, J., Klingner, J., Hanrahan, P., and Tversky, B. (2003). "Designing Effective Step-by-Step Assembly Instructions." Proceedings of ACM SIGGRAPH 2003, ACM Transactions on Graphics (TOG), Vol. 22 No. 3, 828-837.

Aho, Alfred & E. Hopcroft, John & Ullman, Jeffrey. (1987). *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

Akin Ö. (2002), *Variants in Design Cognition. Design Knowing and Learning Cognition in Design Education*. C., Eastman, M. McCracken, and W. Newstetter. Amsterdam, Elsevier: 105–124.

Alexander, C., Ishikawa, S., and Silverstein, M. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York.

Alexander, C. (1965) A City is Not a Tree. *Architectural Forum*, Vol 122, No 1, April 1965, pp 58-62

Andres Sevtsuk and Michael Mekonnen. 2012. Urban network analysis: a new toolbox for measuring city form in ArcGIS. In Proceedings of the 2012 Symposium on Simulation for Architecture and Urban Design (SimAUD '12). Society for Computer Simulation International, San Diego, CA, USA, Article 18, 10 pages.

Andrienko G., Andrienko N., Jankowski P., Keim D., Kraak M. J., MacEachren A. M., Wrobel, S., *Geovisual analytics for spatial decision support: setting the research agenda*, International Journal of Geographic Information Science, vol. 8, 2007, p.839-857.

Anton, I. and Tănase D., (2016) Informed Geometries. Parametric Modelling and Energy Analysis in Early Stages of Design, Energy Procedia, Volume 85, Pages 9-16, ISSN 1876-6102, https://doi.org/10.1016/j.egypro.2015.12.269.

Armour G. C., Buffa E. S. (1963) *A heuristic algorithm and simulation approach to the relative location of facilities*. In: Management Sci, University of California: Los Angeles, 1963. Retrieved from DOI://abs/10.1287/mnsc.9.2.294

Lindenmayer, Aristid. (1968). *Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs*. Journal of theoretical biology. 18. 300-15. 10.1016/0022-5193(68)90080-5.

Batty, Michael & and Longley, M. (1994). *Fractal Cities - A Geometry of Form and Function*. Academic Press, London.

Batty, M. (2013). *New Science of Cities.* MIT Press, 2013, 520 pp., ISBN 978026201952

Bentley, P. (ed.), 1999, *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisco

Bentley, P. J., Corne, D., 2002, *Creative Evolutionary Systems*, Academic Press, San Diego

Boehm, F., 2002, *Evolutionary Design*, in: Arch 153, Berlin

Buckland, Matthew. (2019). *Programming Game AI by Example*

Caldeira, Ana & Kastenholz, Elisabeth. (2019). Spatiotemporal Tourist Behaviour in Urban Destinations: a framework of analysis.

Calixto, Victor, and Gabriela Celani. 2015. *A Literature Review for Space Planning Optimization Using an Evolutionary Algorithm Approach: 1992- 2014*. Blucher Design Proceedings 2 (3): 662–671.

Cao, Kang. (2013). *Modern urban planning theories. Planning Theory*. 12. 321-323. 10.1177/1473095212451042.

Carlos A. Vanegas, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Paul Waddell. 2012. *Inverse design of urban procedural models*. ACM Trans. Graph. 31, 6, Article 168 (November 2012), 11 pages. DOI: https://doi.org/10.1145/2366145.2366187

C. C. Coolen, A. (1998). *A Beginner's Guide to the Mathematics of Neural Networks*. 10.1007/978-1-4471-3427-5_2.

Chapin FS Weiss SF (1962) *Factors Influencing Land Development. An Urban Studies Research Monograph*, Chapel Hill: Center for Urban and Regional Studies, Institute for Research in Social Science, University of North Carolina

Christopher Alexander, Sara Ishikawa, Murray Silverstein (1977). *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press. 0-19-501919-9

Cláudia Maria de Almeida, Michael Batty, Antonio Miguel Vieira Monteiro, Gilberto Câmara, Britaldo Silveira Soares-Filho, Gustavo Coutinho Cerqueira, Cássio Lopes Pennachin (2003). *Stochastic cellular Automata modeling of urban land use dynamics: empirical development and estimation*. Computers, Environment and Urban Systems, 481-509.

Clevenger, Caroline & Haymaker, John. (2011). *Metrics to assess design guidance. Design Studies* - DESIGN STUD. 32. 431-456. 10.1016/j.destud.2011.02.001.

Cook, P (ed.), Archigram, New York: Praeger, 1973

Collins George, "Linear Planning throughout the world", Journal of the Society of Architectural Historians, Vol. 18, October 1959

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*, Third Edition (3rd ed.). The MIT Press.

Dang, Nguyet & Kawasaki, Akiyuki. (2016). *A Review of Methodological Integration in Land-Use Change Models*. International Journal of Agricultural and Environmental Information Systems. 7. 1-25. 10.4018/IJAEIS.2016040101.

Dangermond, J. *Can geodesign help us adapt to climate change?*. Retrieved from Esri Insider blog, 23 April 2012

Daniel G. Aliaga, Carlos A. Vanegas, and Bedrich Benes. 2008. *Interactive example-based urban layout synthesis*. ACM Trans. Graph. 27, 5, Article 160 (December 2008), 10 pages. DOI: https://doi.org/10.1145/1409060.1409113

Dmitriev, V & S. Kurkina, E & E. Simakova, O. (2011). *Mathematical models of urban growth. Computational Mathematics and Modeling.* 22. 54-68. 10.1007/s10598-011-9088-8.

Eastman, Chuck & Teicholz, Paul & Sacks, Rafael & Liston, Kathleen. (2008). BIM for the Construction Industry. 10.1002/9780470261309.ch6.

Edward A. Bender, S. Gill Williamson, and Mathematics. 2011. *Mathematics for Algorithm and Systems Analysis*. Dover Publications, Inc., New York, NY, USA.

Ervin, S., *A system for geodesign*, International Conference on Digital Landscape Architecture, 25-31 May, 2011, Dessau, Germany.

Eugénio Rodrigues, Adélio Rodrigues Gaspar, Álvaro Gomes, *an evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture*, Part 2: Validation and performance tests, Computer-Aided Design, Volume 45, Issue 5, 2013, Pages 898-910, ISSN 0010-4485

Finn V. Jensen. 1996. Introduction to Bayesian Networks (1st ed.). Springer-Verlag, Berlin, Heidelberg.

Flaxman, M., Buhmann, E., Peitsch, M., Krtzler, E. *Fundamentals of Geodesign*, Proceedings of Digital Landscape Architecture, 26-29, May, 2010, Aschersleben, Germany.

Flager F. and J. Haymaker (2007). *A Comparison of Multidisciplinary Design, Analysis and Optimization*, Processes in the Building Construction and Aerospace Industries. *24th International Conference on Information Technology in Construction*. I. Smith. Maribor, Slovenia: 625-630

Flager F, Welle B, Bansal P, Soremekun G, Haymaker J (2009). Multidisciplinary process integration and design optimization of a classroom building, ITcon Vol. 14, pg. 595-612, https://www.itcon.org/2009/38

Flemming U (1987) The Role of Shape Grammars in the Analysis and Creation of Designs, in Computability of Design, Y.E. Kalay, Editor, Wiley: New York, 245-272

Franco P. Preparata and Michael I. Shamos. 1985. *Computational Geometry: an Introduction.* Springer-Verlag, Berlin, Heidelberg.

Francois Chollet. 2017. *Deep Learning with Python* (1st ed.). Manning Publications Co., Greenwich, CT, USA.

Friedman Y, *Toward a Scientific Architecture*, Cambridge, Mass, 1975

Freedman, Ora and Kern, Clifford R., (1997), A model of workplace and residence choice in two-worker households, *Regional Science and Urban Economics*, **27**, issue 3, p. 241-260

Galle, Per. (1981). *An Algorithm for Exhaustive Generation of Building Floor Plans*. Communications of the ACM December 1981. vol. 2: pp.813-823, [3]: ill. includes bibliography. 24. 10.1145/358800.358804.

Guoning Chen, Gregory Esch, Peter Wonka, Pascal Muller, and Eugene Zhang. 2008. *Interactive procedural street modeling*. ACM Trans. Graph. 27, 3, Article 103 (August 2008), 10 pages. DOI: https://doi.org/10.1145/1360612.1360702

Gero, J.S., Mt Brazier, F., 2002, *Agents in Design, Key Center of Design and Computation*, University of Sydney, Sidney, Australia.

Gershenfeld, N., 1999, *The Nature of Mathematical Modeling*, Cambridge University Press, New York

Grasl, Thomas & Economou, Athanassios. (2010). *Palladian Graphs: Using a graph grammar to automate the Palladian grammar*. FUTURE CITIES [28th eCAADe Conference Proceedings / ISBN 978-0-9541183-9-6] ETH Zurich (Switzerland) 15-18 September 2010, pp.275-283.

Gurr J.M., Walloth C. (2014) Introduction: Towards a Transdisciplinary Understanding of Complex Urban Systems. In: Walloth C., Gurr J., Schmidt J. (eds) Understanding Complex Urban Systems: Multidisciplinary Approaches to Modeling. Understanding Complex Systems. Springer, Cham

Haddad, M. A. (2015). *Review: A framework for geodesign: Changing geography by design*. Journal of Planning, Education and Research June, 35(2), 228-230.

Hearn, D., Baker, P., 1997, *Computer Graphics*, Prentice Hall, Upper Saddle River, NJ, 1997

Heppenstall, A.J. & Crooks, Andrew & See, Linda & Batty, Michael. (2012). *Agent-Based Models of Geographical Systems*. 10.1007/978-90-481-8927-4.

Herbert A. Simon. 1996. The Sciences of the Artificial (3rd Ed.). MIT Press, Cambridge, MA, USA.

Homayouni,H.(2007). A Genetic Algorithm Approach to Space Layout Planning Optimization.thesis. University of Washington. Retrieved from http://citeseerx.ist.psu.edu/ viewdoc/summary?doi=10.1.1.114.7729

Geddes Patrick, Cities in Evolution, New York: Howard and Fertig, 1968 (orig. 1915)

Goodfellow I, Bengio Y, and Courville A, 2016, *Deep learning*, The MIT Press, 800 pp, ISBN: 0262035618

Hartmann, T., Fischer, M., and Haymaker, J. (2009). "Implementing Information Systems with Project Teams Using Ethnographic–Action Research." Advanced Engineering Informatics, 23(1), 57-67.

Haymaker, J. (2006). "Communicating, Integrating, and Improving Multidisciplinary Design Narratives." International Conference on Design Computing and Cognition 2006, Springer, Netherlands, 635-653.

Haymaker, J. and Chachere, J. (2006). "Coordinating Goals, Preferences, Options, and Analyses for the Stanford Living Laboratory Feasibility Study." Intelligent Computing in Engineering and Architecture 13th EG-ICE Revised Selected Papers, Lecture Notes in Computer Science, Vol. 4200/2006, Ian Smith (ed.), Springer-Verlag, Berlin, Heidelberg, New York, 320-327.

Haymaker, J., Ayaz, E., Fischer, M., Kam, C., Kunz, J., Ramsey, M., Suter, B., and Toledo, M. (2006). "Managing and Communicating Information on the Stanford Living Laboratory Feasibility Study." Journal of Information Technology in Construction, Vol. 11, 607-626.

Haymaker, J., Fischer, M., Kunz, J., and Suter, B. (2004). "Engineering Test Cases to Motivate the Formalization of an AEC Project Model as a Directed Acyclic Graph of Views and Dependencies." Journal of Information Technology in Construction, Vol. 9, 419-41.

Haymaker J., Kunz, J., Suter, B., and Fischer, M. (2004). "Perspectors: Composable, Reusable Reasoning Modules To Construct an Engineering View from Other Engineering Views." Advanced Engineering Informatics, Vol. 18/1, 49-67.

Haymaker, J. R., Keel, P., Ackermann, E., and Porter, W. (2000). "Filter Mediated Design: Generating Coherence in Collaborative Design." Design Studies, Vol. 21, No. 2, Elsevier Science Ltd., 205-220.

Homayouni, H.(2007). A Genetic Algorithm Approach to Space Layout Planning Optimization.thesis. University of Washington. Retrieved from http://citeseerx.ist.psu.edu/ viewdoc/summary?doi=10.1.1.114.7729

Hooman Foroughmand Araabi (2014) The Urban Design Reader (second edition), Journal of Urban Design, 19:2, 266-267, DOI: 10.1080/13574809.2013.853248

Iacono, Michael & Levinson, David & El-Geneidy, Ahmed & Rania, Wasfi. (2012). Markov Chain Model of Land Use Change in the Twin Cities. TeMA: Journal of Land Use, Mobility and Environment. 8. 10.6092/1970-9870/2985.

James Rumbaugh, Ivar Jacobson, and Grady Booch. 2004. *Unified Modeling Language Reference Manual*, the (2nd Edition). Pearson Higher Education.

Jabi, Wassim. (2014). Parametric Spatial Models for Energy Analysis in the Early Design Stages. Simulation Series. 46.

Jane Jacobs (1961). *The Death and Life of Great American Cities*. Vintage Books

Jo, Jun & Gero, John. (1998). Space layout planning using an evolutionary approach. Artificial Intelligence in Engineering. 12. 149-162. 10.1016/S0954-1810(97)00037-X.

John W. Dyckman (1961) Planning and Decision Theory, Journal of the American Institute of Planners, 27:4, 335-345, DOI: 10.1080/01944366108978368

Kenneth H. Rosen. 2002. *Discrete Mathematics and its Applications* (5th ed.). McGraw-Hill Higher Education.

Kevin Lynch & Lloyd Rodwin (1958) A Theory of Urban Form, Journal of the American Institute of Planners, 24:4, 201-214, DOI: 10.1080/01944365808978281

Lee, Douglass. (1973). *Requiem for Large-Scale Urban Models*. Journal of the American Institute of Planners 39, pp. 163-178. 39. 10.1080/01944367308977851.

Leontief, Wassily. (1966). *Input-output economics.* New York: Oxford University Press.

Lopes, Ricardo, Tim Tutenel, Ruben Michaël Smelik, Klaas Jan de Kraker and Rafael Bidarra. *A Constrained Growth Method for Procedural Floor Plan Generation* (2010).

Luisa Gama Caldas, Leslie K Norford, *A design optimization tool based on a genetic algorithm*, Automation in Construction, Volume 11, Issue 2, 2002, Pages 173-184, ISSN 0926-5805

Lloyd Rodwin, *Nations & Cities*, Boston: Houghton Mifflin, 1970

Lynch, Kevin. *A Theory of Good City Form*. Cambridge, Mass.: MIT Press, 1981.

March, Lionel (1976). *The Architecture of Form*. Cambridge, UK: Cambridge University Press.

March, Lionel; Steadman, Philip (1974). *The Geometry of Environment: An Introduction to Spatial Organization in Design*. Cambridge, MA: MIT Press.

March, Lionel; Martin, Leslie (1972). *Urban Space and Structures*. Cambridge, UK: Cambridge University Press.

Martin, L. (2000). *The grid as generator*. Architectural Research Quarterly, 4(4), 309-322. doi:10.1017/S1359135500000403

Miao, Yufan & Koenig, Reinhard & Knecht, Katja & Konieva, Kateryna & Buš, Peter & Chang, Mei-Chih. (2018). Computational urban design prototyping: Interactive planning synthesis methods—a case study in Cape Town. International Journal of Architectural Computing. 16. 212-226. 10.1177/1478077118798395.

McHarg, Ian L. 1969. *Design with nature*. Garden City, N.Y.: Published for the American Museum of Natural History, The Natural History Press.

M. Torrens, Paul & Benenson, Itzhak. (2005). *Geographic automata systems*. Int J Geogr Inf Sci. International Journal of Geographical Information Science. 19. 385-412. 10.1080/13658810512331325139.

McElvaney, Shannon. *Geodesign : Case Studies in Regional and Urban Planning*. 1st ed. Redlands, Calif.: Environmental Systems Research Institute, 2012.

Michael Batty. 2013. The New Science of Cities. The MIT Press.

Michalek, Jeremy & Choudhary, R & Papalambros, Panos. (2002). *Architectural layout design optimization*. Engineering Optimization. 34. 461-484. 10.1080/03052150214016.

Michael J. Laszlo. 1995. *Computational Geometry and Computer Graphics in C++*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Michael McMillan. 2014. *Data Structures and Algorithms with Javascript*. O'Reilly Media, Inc.

Minsky, M., *The Society of Mind*, Simon & Schuster, New York, NY, 1987.

Mitchell W.J., *The theoretical foundation of computer-aided architectural design*, in: Environment and Planning B, Volume 2, 1975, pp. 127-150.

Mitchell, W.J., *The Logic of Architecture*, MIT Press, Cambridge, Massachusetts, 1990.

Mumford Lewis, *The City in History*, New York: Harcourt Brace & World, 1961

Nagasaka, Ichiro. (2020). Syntax and Semantics of Pattern Language.

Negroponte, N., *Being Digital*, Vintage Books, New York, 1995.

Nauata, Nelson & Chang, Kai-Hung & Cheng, Chin-Yi & Mori, Greg & Furukawa, Yasutaka. (2020). *House-GAN: Relational Generative Adversarial Networks for Graph-constrained House Layout Generation.*

Nuaimi, E., Al Neyadi, H., Mohamed, N. et al. Applications of big data to smart cities. J Internet Serv Appl 6, 25 (2015). DOI:10.1186/s13174-015-0041-5

Yunhe Pan, Yun Tian, Xiaolong Liu, Dedao Gu, Gang Hua, Urban Big Data and the Development of City Intelligence, Engineering, Volume 2, Issue 2, 2016, Pages 171-178, ISSN 2095-8099, https://doi.org/10.1016/J.ENG.2016.02.003.

Parish Y.I.H. & Mulller P. (2001). *Procedural Modelling of Cities*. Proceedings of SIGGRAPH. 2001. 301-308. 10.1145/1185657.1185716.

Peng et al., C.H. Peng, Y.L. Yang, P. Wonka, *Computing layouts with deformable templates*, ACM Trans. Graph, 33 (4) (2014), pp. 99-110

Putman, S. H. (1983). *Policy analysis of transportation and land use.* London: Pion Ltd.

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning* (1st ed.). MIT Press, Cambridge, MA, USA.

Roozenburg N.F.M., *Cross N.G., Models of the design process: integrating across the disciplines*, Design Studies, Volume 12, Issue 4, 1991, Pages 215-220, ISSN 0142-694X, https://doi.org/10.1016/0142-694X(91)90034-T.

Rosenman M.A. (1996) *The Generation of Form Using an Evolutionary Approach*. In: Gero J.S., Sudweeks F. (eds) Artificial Intelligence in Design '96. Springer, Dordrecht

Salingaros, Nikos A. 2006. *A Theory of Architecture*. Solingen: Umbau-Verl.

Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh Vazirani. 2006. *Algorithms* (1 ed.). McGraw-Hill, Inc., New York, NY, USA.

Schultz R. (2014) Uncertainty in Urban Systems: How to Optimize Decision Making Using Stochastic Programming. In: Walloth C., Gurr J., Schmidt J (eds) Understanding Complex Urban Systems: Multidisciplinary Approaches to Modelling. Understanding Complex Systems. Springer, Cham.

Simon, H., The Science of the Artificial, MIT Press, Cambridge, Massachusetts, 1996.

Straszhem, Mahlon, (1987), *The theory of urban residential location*, ch. 18, p. 717-757 in Mills, E. S. eds., Handbook of Regional and Urban Economics, vol. 2, Elsevier.

Steadman, P., Bruhns, H. R., Holtier, S., Gakovic, B., Rickaby, P. A., & Brown, F. E. (2000). *A Classification of Built Forms*. Environment and Planning B: Planning and Design, 27(1), 73–91. https://doi.org/10.1068/bst7

Steadman, P. (2014). *Building types and built forms*.

Steinitz, C. (1995). *Design is a verb; Design is a noun*. Landscape Journal, 14(2). 188-200.

Steinitz, C., (2008) *On Scale and complexity and the need for spatial analysis,* Specialist Meeting on Spatial Concepts in GIS and Design, 15-16, Santa Barbara, California.

Steinitz C., (2008) *Landscape planning: A brief history of influential ideas*, Journal of Landscape Architecture, 3:1, 68-74, DOI: 10.1080/18626033.2008.9723397

Steinitz, Carl. *A Framework for Geodesign: Changing Geography by Design*. First ed. Redlands, Calf.: Esri, 2012.

S. Liggett, Robin & J. Mitchell, William. (1981). *Optimal space planning in practice*. Computer-Aided Design. 13. 277-288. 10.1016/0010-4485(81)90317-1.

Steadman, Philip. *The Evolution of Designs: Biological Analogy in Architecture and The Applied Arts*. London: Routledge, 2008. International Congress on Genetics (1932), 355–366.

Jeffrey Star & John Estes (1991) *Geographic information systems: An introduction*, Geocarto International, 6:1, 46, Doi: 10.1080/10106049109354297

Steven S. Skiena. 2008. *The Algorithm Design Manual* (2nd ed.). Springer Publishing Company, Incorporated.

Strang, Gilbert. *Linear algebra and its applications*. Belmont, CA: Thomson, Brooks/Cole, 2006.

Stretton Hugh*, Ideas for Australian Cities*. Melbourne: Gregorian House, 1971

Stuart Russell and Peter Norvig. 2009. Artificial Intelligence: A Modern Approach (3rd. ed.). Prentice Hall Press, USA.

Stiny G (1975) *Pictorial and formal aspects of shape and shape grammars*. Interdisciplinary Systems Research, Birkhäuser

Stiny G (2006) *Shape: talking about seeing and doing*, MIT Press

Terzidis, Kostas. 2006. *Algorithmic Architecture*. Oxford; Burlington, MA: Architectural Press.

Thomas Courtat, Catherine Gloaugen, Stephane Duoady (2011). *Mathematics and Morphogenesis of Cities; A Geometrical Approach*. Physical review. E, Statistical, nonlinear, and soft matter physics 83 3 Pt 2 (2011): 036106.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition* (3rd ed.). The MIT Press.

Ullman, Shimon. 2000. *High-Level Vision: Object Recognition and Visual Cognition*. MIT Press.

Victor Gane, John Haymaker, *Design Scenarios: Enabling transparent parametric design spaces*, Advanced Engineering Informatics, v.26 n.3, p.618-640, August 2012 DOI://10.1016/j.aei.2012.04.008

Walpole, Ronald E., Raymond H. Myers, Sharon L. Myers, and Keying Ye. *Probability & Statistics for Engineers & Scientists*. 9th edition. Boston: Prentice Hall, 2012.

Wayne L. Winston. 2003. *Introduction to Mathematical Programming: Applications and Algorithms*. Duxbury Resource Center.

William J. Mitchell, *Computer-Aided Architectural Design*, John Wiley & Sons, Inc., New York, NY, 1977

Yang PPJ (2012) *Complexity Question in Urban Systems Design*. J Archit Eng Tech 1: e107. Doi: 10.4172/2168-9717.1000e107

Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D. Goodman, and Pat Hanrahan. 2012. *Synthesizing open worlds with constraints using locally annealed reversible jump MCMC*. ACM Trans. Graph. 31, 4, Article 56 (July 2012), 11 pages. DOI: https://doi.org/10.1145/2185520.2185552

Yong-Liang Yang, Jun Wang, Etienne Vouga, Peter Wonka*, Urban pattern: layout design by hierarchical domain splitting*, ACM Transactions on Graphics (TOG), v.32 n.6, November 2013 DOI://10.1145/2508363.2508405

Wilson, AG; (2009) *The 'thermodynamics' of the city*. In: Reggiani, A and Nijkamp, P, (eds.) Complexity and spatial networks. (pp. 11-31). Springer: Berlin.

Woodbury, Robert. 2010. *Elements of Parametric Design*. London; New York: Routledge.

M. Rocke, D. (2000). *Genetic Algorithms + Data Structures = Evolution Programs by Z. Michalewicz.* Journal of the American Statistical Association. 95. 347-348. 10.2307/2669583.

Zifeng Guo, Biao Li, *Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system*, Frontiers of Architectural Research, Volume 6, Issue 1, 2017, Pages 53-62, ISSN 2095-2635.